НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:

Завідувач кафедри

_____ Олександр Коваль

«___»_____2020 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інформаційні технології моніторингу довкілля»

спеціальності 122 «Комп'ютерні науки та інформаційні технології»

на тему: «Програмні засоби збору, накопичення та WEB відображення даних вимірювань в інженерних мережах»

Виконав:
студент IV курсу, групи ТМ-62
Батін Олексій Олександрович _____

Керівник:
асистент
Швайко Валерій Григоворич _____

Рецензент:
Головний енергоменеджер НТУУ "КПІ" ім. Ігоря Сікорського, к.т.н
Шевченко Олена Миколаївна _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2020 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки 122 Комп’ютерні науки та інформаційні технології

Спеціалізація Інформаційні технології моніторингу довкілля

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр Коваль
(підпис)
”___”_____2020р.

ЗАВДАННЯ

на дипломну роботу студенту

Батіну Олесію Олександровичу
(прізвище, ім’я, по батькові)

1. Тема роботи “Програмні засоби збору, накопичення та WEB відображення даних вимірювань в інженерних мережах”

керівник роботи Асистент Швайко Валерій Григорович
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”25” травня 2020р. № **1168-с**

2. Строк подання студентом роботи _____
3. Вихідні дані до роботи Node.js, TypeScript, Javascript, Nest.js

4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) проаналізувати існуючі програмні рішення для збору, накопичення та WEB відображення даних вимірювань в інженерних мережах, розробити програмний продукт, який надає користувачу веб-інтерфейс для перегляду даних вимірювань датчиків та коректну передачу даних.

5. Перелік ілюстративного матеріалу

1. Актуальність 2. Мета та завдання 3. Потенційні користувачі 4. Огляд існуючих рішень. 5. Технологія передачі даних – LORAWAN 6. Високорівневе представлення системи 7. Відображення інженерних мереж 8. Діалогове вікно 9. Перегляд таблиці з даними вимірювань 10. Перегляд атрибутів таблиці 11. Використання даних з таблиці 12. Висновки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання ”___”_____201___ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	08.02.20	
2.	Вивчення та аналіз задачі	13.04.20-19.04.20	
3.	Розробка архітектури та загальної структури системи	20.04.20-26.04.20	
4.	Розробка структур окремих підсистем	27.04.20-03.05.20	
5.	Програмна реалізація системи	04.05.20-13.05.20	
6.	Оформлення пояснювальної записки	14.05.20-16.05.20	
7.	Захист програмного продукту	17.05.20	
8.	Передзахист	09.06.20	
9.	Захист	17.06.20	

Студент _____ _____Олексій БАТІН_____
(підпис) (прізвище та ініціали,)

Керівник роботи _____ _____Валерій ШВАЙКО_____
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Метою дипломної роботи є створення системи, що надає користувачам програмні засоби для збору та взаємодії з даними вимірювань в інженерних мережах. Серверний додаток було створено на платформі Node.js з використанням мови програмування TypeScript. Для відображення даних у WEB інтерфейсі використовується хмарне рішення ArcGIS Online, що надає користувачам інтерфейс для взаємодії з картою та геоінформаційною базою даних.

Розроблений програмний продукт може бути використаний у сферах моніторингу та аналізу енергоефективності будівель, в яких встановлено розумні лічильники.

Пояснювальна записка містить 60 сторінок, включає 19 рисунків, 3 таблиці та 13 посилань.

Ключові слова: геоінформаційна система, Node.js, JavaScript, TypeScript, Nest.js, IoT, розумний лічильник, LoRaWAN, ArcGIS Online.

ABSTRACT

The purpose of the diploma thesis is to create a system that provides users software tools for collecting and interacting with measurement data in engineering networks. A server application created on the Node.js platform using the TypeScript programming language. The cloud solution ArcGIS Online is used to display data in the WEB interface which provides users interface for interaction with the map and geographic information database.

The developed software product can be used in areas such as monitoring and analysis of energy efficiency in buildings, where smart meters are installed.

The explanatory note contains X pages, including 19 figures, 3 tables and 13 references.

Keywords: geographic information system, Node.js, JavaScript, TypeScript, Nest.js, IoT, smart meter, LoRaWAN, ArcGIS Online.

ЗМІСТ

ВСТУП

Поняття Інтернету речей (IoT) представляє собою наступний рівень розвитку Інтернету. Він направлений на те, щоб дати можливість будь-якому пристрою змогу для обміну інформації з іншими об’єктами, за допомогою мережі Інтернет, такими об’єктами можуть бути автомобілі, будівлі, тощо. В сфері Інтернету речей було проведено декілька дослідів, які призвели до його популярності та активному розвитку. Сьогодні проекти Інтернету речей мають свої категорії і таким чином різні системи можна класифікувати по різному, прикладами таких категорій є “Розумний будинок”, “Розумне місто”, “Розумний транспорт”. Також не винятком є ситуації коли рішення проблеми в одній із цих категорій є настільки вдалим та може слугувати для вирішення проблеми в іншій класифікації, тому для всіх цих систем зазвичай використовуються однакові архітектурі рішення [1].

У наш час розвиток та вдосконалення існуючих систем підштовхує людей до автоматизації багатьох задач. Так, за допомогою системи “Розумний будок”, люди змогли перекласти низку обов’язків на комп’ютер за допомогою якого відбувається контроль та регуляція основних процесів, що надають змогу власнику житла не замислюватись над причинами та діями, які потрібно зробити для поліпшення температурного режиму, вологості та енергозабезпечення свого житла.

Інтеграція інтелектуальних вимірюваних приладів у місті за допомогою Інтернету речей дозволяє робити сбір всіх даних та їх аналіз, таким чином бути невід’ємним компонентом в Інтернеті речей. Вимірювальні прилади та датчики є основоположною частиною для підтримки зв’язку між різними системами, бо на основі аналізу значень з приладів система повинна реагувати та забезпечувати правильну роботу. Інтеграція інформаційної системи побудованої на розумних лічильниках, дозволяє контролювати інші підсистеми, що можуть реагувати та аналізувати ці дані.

Встановлення сучасної системи, що стосується вимірювань, комунікації та автоматизації сприяє розвитку міста. Мета встановлення таких систем, полягає в тому, щоб забезпечити та досягти кращого управління мережами електроенергії, води, газу та ефективного балансу між попитом та споживання цих ресурсів. Тому, саме розумний лічильник є ключовим компонентом у цій системі, яка може бути інтегрована з Інтернетом речей.

Для зручного відображення, аналізу та фіксації географічної інформації використовуються геоінформаційні системи, що дозволяють поєднувати модельне зображення території з інформацією, яку можна представити у табличному виді [2].

Метою даної роботи є створення єдиної системи, що дозволяє у зручному вигляді робити ефективне управління та відображення даних з розумних лічильників електроенергії, води та газу.

Для створення системи відображення даних вимірювань з розумних лічильників в інженерних мережах потрібно зробити систему, яка наддасть змогу об’єднати існуючу геоінформаційну систему з системою, що здійснює збір даних з розумних лічильників у різних корпусах. Для доступу та перегляду геоінформаційних даних було обрано рішення ArcGIS Online, а для доступу та налаштування розумних лічильників, використовується програмний продукт Actility ThingPark Wireless.

Розроблена система повинна забезпечувати правильність та коректну передачу даних між двома системами, що не мають змогу взаємодіяти між собою.

1. ЗАДАЧА РОЗРОБКИ СИСТЕМИ ДЛЯ ЗБОРУ, НАКОПИЧЕННЮ ТА WEB ВІДОБРАЖЕННЯ ДАНИХ ВИМІРЮВАНЬ В ІНЖЕНЕРНИХ МЕРЕЖАХ

Система для збору, накопиченню та відображення даних вимірювань з розумних лічильників в інженерних мережах надає можливість користувачу системи засоби для зручного аналізу даних. Користувач системи має змогу вносити дані та переглядати інформацію про розумні лічильники та їх показники за період часу. Сервіс ArcGIS Online надає доступ до веб інтерфейсу, де користувач може в інтерактивному режимі побачити місцезнаходження розумних лічильників та інформацію, яка зв'язана з ним. Такою інформацією може бути загальні відомості та характеристика розумного лічильника, а також вимірювання за певний період часу. Для правильного відображення користувач повинен правильно заповнити данні та необхідні атрибути розумного лічильника до системи, після цього дані з цього розумного лічильника будуть автоматично оновлюватись, коли лічильник відправляє дані про свої вимірювання.

Завдяки графічному відображенню розумних лічильників на мапі користувач системи має більше детальне розуміння та його точне географічне розташування.[3]

У розділі розкрито сутність системи, мета створення та основні задачі, які було вирішено в процесі створення інформаційної системи. Також описані основні вимоги до інформаційної системи і інформацію про її потенційних користувачів.

1.1 Мета створення системи

Метою цієї роботи є надання можливостей для зручного та зрозумілого відображення даних вимірювань з розумних лічильників НТУУ "КПІ" на мапі з можливістю управління цими даними.

Основні етапи роботи з розробки такої системи:

- аналіз системи, що здійснює управління розумними лічильниками;
- створення геоінформаційної системи в ArcGIS Online;
- наповнення геоінформаціної бази даних даними про існуючі розумні лічильники;
- розробка програмного продукту, що надає функціонал для синхронізації та оновлення даних між ArcGIS Online та системи управління розумними лічильниками.

Програмний продукт також дозволяє виводити дані з лічильників на карті у web-інтерфейсі, що дозволяє переглядати інформацію та її редагувати з будь-якого пристрою.

1.2 Задачі поставлені перед інформаційною системою

Задачі, які дана система вирішує:

- збір даних з розумних лічильників на території НТУУ “КПІ”;
- реєстрація нових розумних лічильників на території НТУУ “КПІ”;
- відображення даних з розумних лічильників та їх показники;
- управління та редагування інформації за допомогою зручного інтерфейсу.

Вирішення всіх задач потребувало аналіз програмних продуктів, що вже використовувались для обробки інформації такого виду, а також вивчення документації лічильників та системи Actility ThingPark Wireless з якою вони взаємодіють. Для кращого розуміння принципу побудови геоінформаційних систем та особливості їх побудови в ArcGIS Online було використано документацію ArcGIS.

1.3 Вхідні дані

Розроблений програмний продукт має містити інформацію стосовно основних відомостей по розумним лічильникам, а також їх показники. Відповідно, вхідними даними для цієї системи є реальне положення розумних лічильників, що передається у вихідних пакетах разом з інформацією про основні показники [3].

Від користувача вхідними даними є інформація про розумний лічильник та його характеристики. Також користувач може завантажити власну геоінформаційну базу даних, яка буде відображати додаткову інформацію на мапі [4].

Мінімальна кількість вхідних даних, які потрібно вводити користувачу робить систему інтуїтивно зрозумілою та зручною для подальшої модернізації.

1.4 Основні компоненти системи

Для розробки поставлених перед інформаційною системою задач було виділено наступні компоненти:

- web-інтерфейс з конфігурацією ArcGIS Online;
- підсистема передачі даних з розумних лічильників;
- підсистема для накопичення даних у форматі геоінформаційної бази даних;
- підсистема управління розумними лічильниками.

Розроблена система з передачі даних з розумних лічильників надає користувачу можливість відображення цих даних у web-інтерфейсі в зручному вигляді.

Web-інтерфейс взаємодіє із підсистемою накопичення даних та підсистемою передачі даних, за допомогою якої можливий доступ до показників розумних лічильників. Таким чином користувач має змогу переглянути інформацію, які дані були

відправлені розумним лічильником у систему. Також надає доступ до перегляду інформації про розташування розумних лічильників та їх основні атрибути.

Підсистема управління розумними лічильниками допомагає в налаштуванні та коректній роботі повідомлень, що надходять від користувача. Ця система використовує програмне забезпечення Actility ThingPark Wireless, що дозволяє користувачу робити управління датчиками та їх налаштування. Розумні лічильники збирають дані показників та насилають дані до підсистеми передачі даних у вигляді повідомлень з зашифрованим полем, що містить інформацію про показники, місцеположення, час, тощо [5].

Підсистема передачі даних приймає зашифроване повідомлення від датчиків та формує правильну структуру даних, яка буде надіслана до ArcGIS Online. У разі появи нових джерел, що зберігають або надають доступ до показників розумних лічильників, ця система повинна бути оновлена та доповнена функцією обробки інформації з нового джерела.

На (рисунку 1.1) зображена діаграма компонентів, що показує потік даних та залежність підсистем між собою.

Рисунок 1.1 – Діаграма компонентів

Для взаємодію між собою різні модулі використовують інтерфейси компонентів, які представлені програмним забезпеченням. Для “Web Application” інтерфейс забезпечується програмним продуктом ArcGIS Online, аналогічно “Smart Meter” забезпечується Actility ThingPark Wireless.

1.5 Потенційні користувачі

Система, що була розроблена може бути використана персоналом НТУУ “КПІ”, щоб мати можливість переглянути деталізовану інформацію про показники лічильників, їх місцезнаходження на мапі та реєстрацію нових лічильників у системі. Система не вимагає спеціальної конфігурації системи для коректної роботи. Зручний web-інтерфейс, який доступний з будь-якого пристрою, надає можливість здійснювати облік лічильників та їх показників дистанційно.

2. АНАЛІЗ ПРОБЛЕМИ СТВОРЕННЯ ГЕОІНФОРМАЦІЙНОЇ СИТЕМИ ДЛЯ УПРАВЛІННЯ РОЗУМНИМИ ЛІЧИЛЬНИКАМИ

Основною задачею для створення геоінформаційної системи є вирішення проблеми зі створення геоінформаційної бази даних та безпечної передачі даних від лічильника до бази даних. Дані, що отримані з різних джерел повинні бути трансформовані у правильну структуру для подальшої зручної роботи користувача з системою.

2.1 Технології передачі даних в мережі Інтернету речей

Комунікація між пристроями є ключовим моментом, щоб створити мережу Інтернет речей. Бездротова комунікація забезпечує переваги мобільності, які надають переваги у простоті додавання нових пристроїв, що здатні підключитися до мережі Інтернет. У наш час, бездротові сенсорні мережі – це одна з найуспішніших технологій, що використовується для розгортання мереж Інтернету речей. Бездротова сенсорна мережа є ключовою частиною для підтримки взаємодії та інтеграції фізичних об’єктів в єдину мережу.

Використання бездротових мереж робить процес розробки та розгортання нових мереж швидким та недорогим. Завдяки активному розвитку бездротового зв’язку було поліпшено енергоефективність та радіус покриття сигналу. Однак, мережа не була позбавлена таких недоліків як обчислювальна спроможність та завантажена смуга зв’язку. Різні сценарії потребують різних властивостей системи, які спроможні подолати проблеми. Для системи Розумного транспорту система повинна надавати мобільність,

система Розумного міста повинна забезпечувати надійну передачу даних на великі відстані.

Сучасні технології бездротових мереж забезпечують передачу даних в різних діапазонах. На (рисунку 2.1) зображено порівняльну характеристику різних бездротових технологій радіозв'язку, для коротких відстаней використовують технології Bluetooth, RFID, NFC. Для великих та середніх відстаней використовують VSAT, LPWAN.

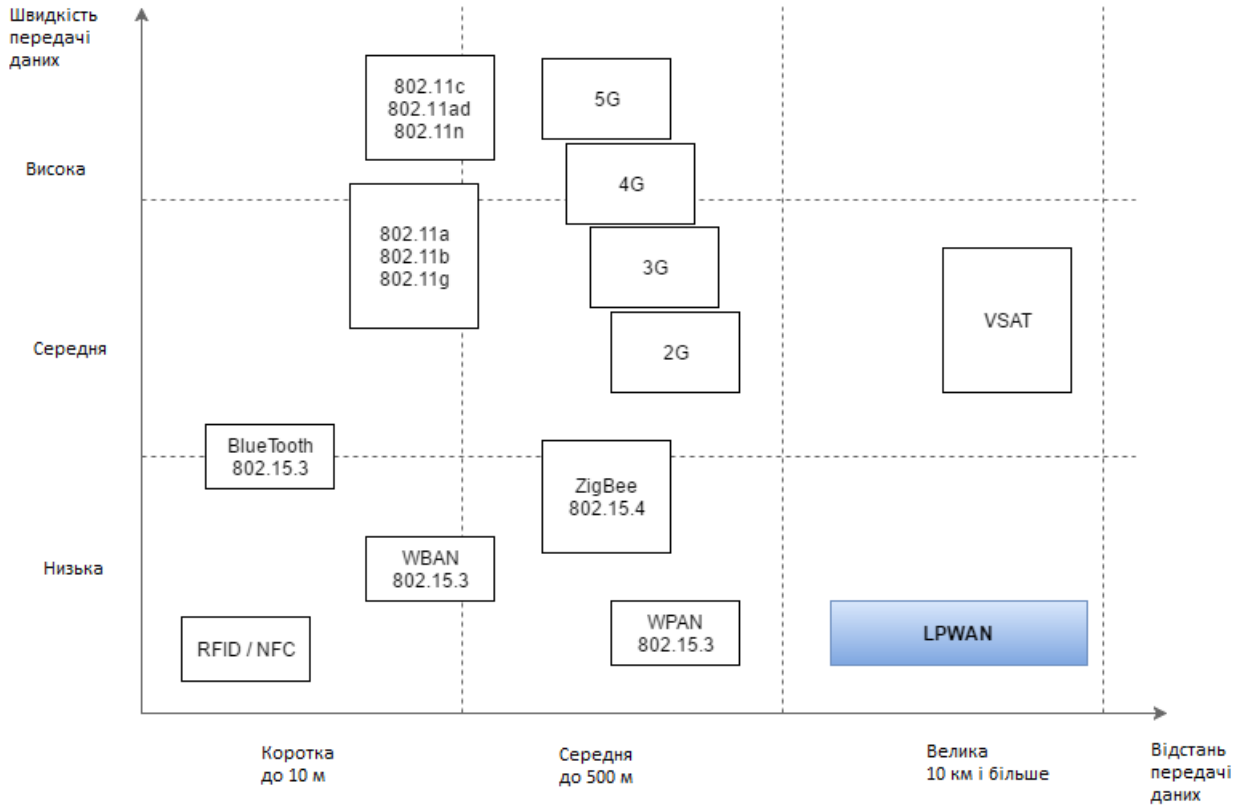

Рисунок 2.1 – Порівняння LPWAN з іншими бездротовими технологіями

Для створення систем, які вимагають низьке енергоспоживання і широке територіальне охоплення, ввели окремий стандарт LPWAN (Low Power Wide Area Netrwork). Низька собівартість модему та найкращі показники енергоефективності, забезпечили популярність LPWAN у розробці мереж для контролю розумних датчиків.

Технології LPWAN можна розділити на дві основні категорії. Це мережі, що потребують ліцензування робочих частот, і мережі, що розгортаються в неліцензованих смугах. Консорціум 3GPP стандартизував два відкриті протоколи LTE-M і NB-IoT, які використовують інфраструктуру оператора. LoRa, SigFox, RPMA і Weightless є прикладами технологій, які працюють у неліцензованих діапазонах і безкоштовно використовують ресурси бездротового спектра для розгортання мереж групової взаємодії й підвищують загальну надійність передавання інформації [6].

2.2 Мережева архітектура LoRaWAN

LoRa – технологія бездротового зв’язку для передачі даних, розроблена для енергоефективних мереж далекого радіуса дії (Low-Power Wide-Area Network або LPWAN), які передбачають автономні запити або передавання даних у пристроях (M2M), що дасть можливість об’єднувати пристрої в мережі на відстань до 15 км за швидкості до 50 Кбіт/с, а також мінімальне споживання електроенергії, що забезпечує до 10 років автономної роботи на одному акумуляторі типу AA [7].

LoRa Alliance – відкрите некомерційне об'єднання, до складу якого входять різні компанії з різних секторів, включаючи технічні компанії, операторів мереж, виробників датчиків, виробників програмного і апаратного забезпечення. Члени LoRa Alliance співпрацюють, щоб забезпечити глобальний успіх протоколу LoRaWAN [8].

На (рисунку 2.2) зображена мережа LoRaWAN, що складається з множини модулів LoRa, які по бездротових з’єднаннях передають дані на один шлюз LoRa Gateway або LoRa Concentrator, або одночасно на декілька, таким чином шлюз і кінцеві пристрої утворюють мережеву топологію типу “зірка”, тому відмова одного з кінцевих пристроїв ніяк не впливає на систему загалом. Зв’язок між шлюзом і кінцевим пристроєм здійснюється з використанням протоколу LoRa. Далі шлюзи, які отримали інформацію, відправляють отримані пакети до мережевого сервера (Network Server), що підключений

за допомогою стандартних технологій з високою швидкістю передачі даних Ethernet, Wi-Fi, 4G. Отримані дані зберігаються та обробляються на сервері аплікацій, користувачі за допомогою клієнтських додатків, встановлених на смартфон або ПК, мають можливість доступу до інформації на сервері додатків, а також застосовувати методи машинного навчання для їх аналізу.

У LoRaWAN мережах передбачено обов'язкове дворівневе шифрування даних двома різними AES-64 і 128 ключами для захисту від несанкціонованого доступу і перехоплення даних, переданих кінцевими пристроями.

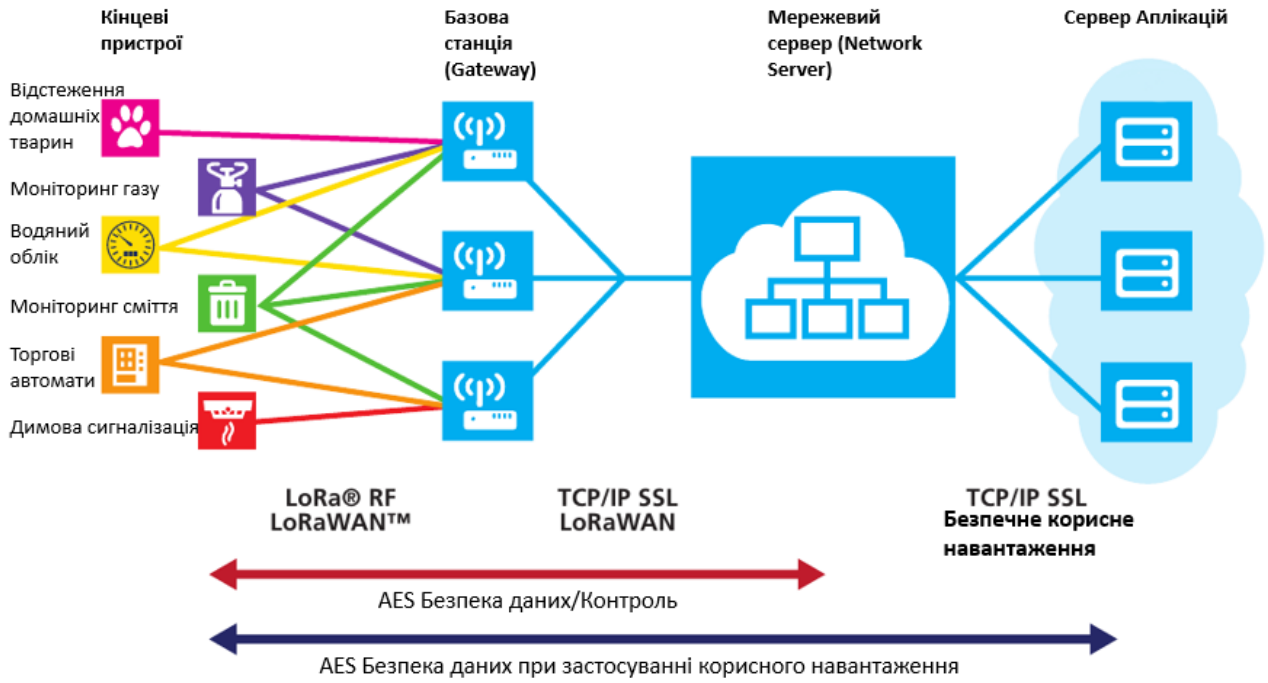

Рисунок 2.2 – Архітектура мережі LoRaWAN

Ключовим елементом мережевого протоколу LoRaWAN є безпека. Його базова структура автентифікації і безпеки заснована на схемі шифрування AES 128, реалізованої в IEEE 802.15.4/2006. Використовуючи окремі ключі для шифрування даних користувача і автентифікації мережі, LoRaWAN пропонує вищий рівень безпеки в порівнянні з реалізаціями з одним ключем. На (рисунку 2.2) зображено два рівня

мережевої безпеки: один для керуючих даних між кінцевими вузлами і мережевим сервером і один для призначених для користувача даних, які транспортуються від кінцевого вузла на сервер додатків. Таким чином, визначені два конкретних 128-бітних зашифрованих ключа AES: мережевий ключ сеансу NwkSKey і ключ сеансу додатка AppSKey. Кожен пристрій IoT матиме свої власні унікальні NwkSKey і AppSKey.

Існує два способи підключення пристрою IoT або так званого кінцевого вузла до мережі LoRaWAN. Перший з яких називається “Активація за допомогою персоналізації” (ABP). У цьому методі NwksKey і AppsKey вже зберігаються в пристрої IoT разом з унікальним 32-бітовим адресою пристрою і унікальним 24-бітовим ідентифікатором мережі, який ідентифікує конкретну мережу LoRaWAN, до якої пристрій призначений для підключення.

Спосіб підключення пристрою IoT до мережі LoRaWAN за замовчуванням використовує процедуру OTAA. У цьому методі кожен пристрій IoT буде відправляти повідомлення запиту на приєднання до мережевого сервера, який потім направляє це повідомлення на сервер об'єднання. Ця команда MAC запиту на приєднання буде містити три поля даних (рисунок 2.3).

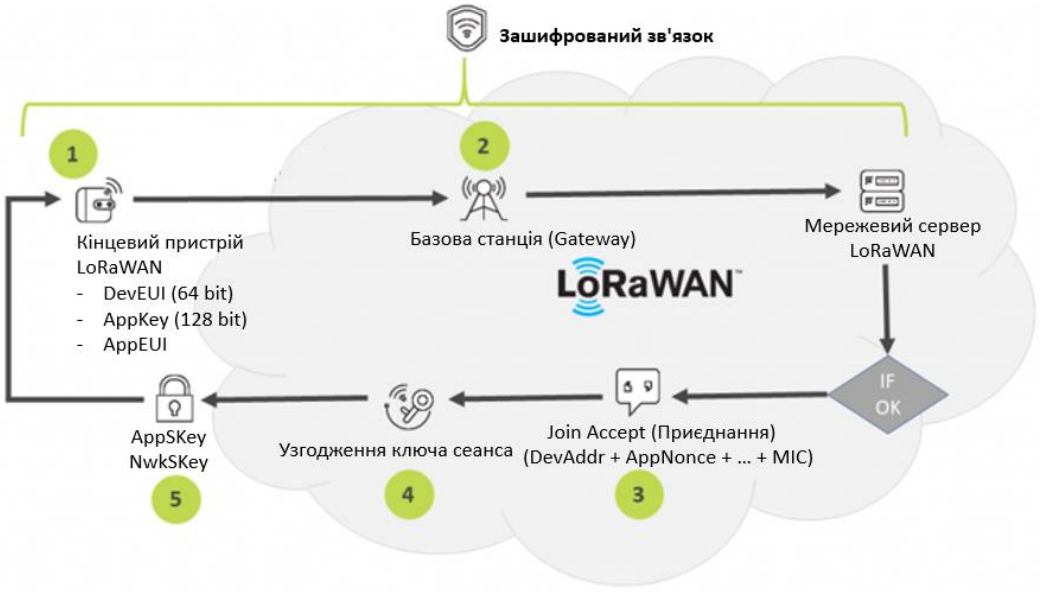

Рисунок 2.3 – Метод підключення OTAA в LoRaWAN

Кожний кінцевий пристрій розгортається з 64-біт DevEUI, 64-біт AppEUI і 128-біт AppKey. DevEUI є унікальним ідентифікатором для пристрою, який має 64-розрядний адрес, що можна порівняти з MAC-адресою для пристрою TCP/IP. AppKey використовується для криптографічного підпису запиту на приєднання, а також AppKey використовується, коли вузол відправляє повідомлення на з'єднання (рисунок 2.3).

Вузол відправляє повідомлення запиту на з'єднання, що складається з його AppEUI і DevEUI. Крім того, він відправляє DevNonce, який є унікальним і випадково згенерованим двобайтовим значенням, використовуваним для запобігання атак. Сервер об'єднання буде зберігати ці випадкові токени пристроїв з попередніх повідомлень про запити на з'єднання від кожного кінцевого вузла.

Активація методом ABP відрізняється від OTAA тим, що вузли поставляються з DevAddr і обома сеансовими ключами NwkSKey і AppSKey, які повинні бути унікальними для вузла. Оскільки вузли вже мають необхідну інформацію і ключі, вони можуть почати зв'язок з сервером без необхідності обміну повідомленнями про приєднання. LoRaWAN при активації через OTAA або ABP забезпечує, що всі майбутні повідомлення будуть зашифровані та підписані з використанням комбінації ключа мережевого сеансу NwkSKey та ключа сеансу додатка AppSKey.

Ці два сеансових ключа NwkSKey і AppSKey є унікальними для кожного пристрою і для кожного сеансу. Якщо пристрій динамічно активується через OTAA, ці ключі відновлюються при кожній активації. Якщо пристрій статично активовано методом ABP, ці параметри залишаються незмінними до тих пір, поки вони не будуть змінені вручну.

Якщо сервер приєднання отримує запит на приєднання в майбутньому від певного кінцевого вузла з токеном пристрою, ідентичним який було недавно отриманого, сервер приєднання ігнорує запит на приєднання. Це запобіжить так звані «атаки повторного відтворення», коли хакер може якимось чином перехопити радіоповідомлення із запитом на приєднання від конкретного кінцевого вузла і відтворити те саме

повідомлення з наміром відключити вихідний кінцевий пристрій від мережі LoRaWAN. На (рисунку 2.4) зображено, які поля використовуються для генерації як AppSKey, так і NwkSKey.

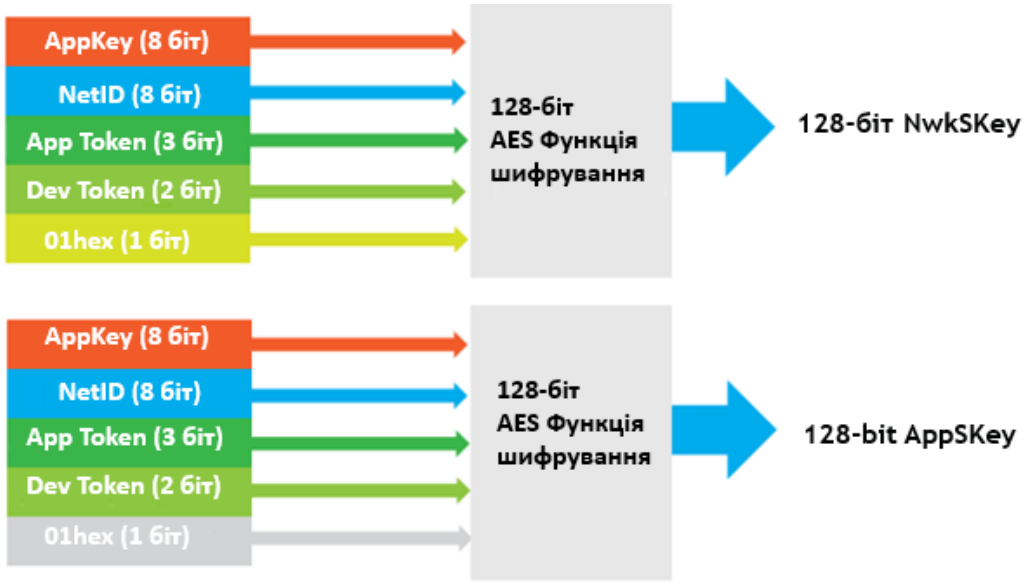

Рисунок 2.4 – Генерація NwkSKey та AppSKey

Для вирішення різних завдань на основі протоколу LoRa передбачено три класи пристроїв, які максимально ефективно використовують функції зв'язку до шлюзу.

На (рисунку 2.5) зображено відповідність між класом та типом датчика, що використовує властивості свого класу для найбільш ефективної взаємодії. Розподіл пристроїв на три класи дозволяє адаптувати систему та збалансувати навантаження на канал обміну.

Всі кінцеві вузли мережі LoRaWAN повинні щонайменше підтримувати клас A. Існує можливий сценарій, коли кінцеві вузли, налаштовані як клас A або клас B, можуть бути тимчасово або на дуже короткий термін налаштовані як пристрої класу C, щоб завжди прослуховувати ефір.

Рисунок 2.5 – Класи кінцевих пристроїв LoRa

Порівняння класів кінцевих пристроїв LoRa:

- Клас A. Пристрої із двостороннім каналом. Зв’язок ініціює кінцевий пристрій, після чого виділяють два часових вікна, протягом яких очікується відповідь від мережі. Цей клас має найвищу енергоефективність, бо пристроям необхідно отримати пакет даних із сервера тільки після відправлення свого пакета даних і найпоширеніший на практиці;
- Клас B. Пристрої з двостороннім каналом. Додатково до функцій пристроїв “класу A” відкривають додаткові вікна приймання за розкладом. Для цього кінцевий пристрій синхронізується за спеціальними сигналами, які отримує від шлюзу. Ініціатором обміну може бути як кінцевий пристрій, так і сервер LoRaWAN мережі. Пристрої класу B також мають всі можливості пристроїв класу A;

- Клас С. Пристрої з двостороннім каналом і максимальним приймальним вікном. Кінцеві пристрої мають майже безперервно відкрите вікно приймання, яке закривається тільки на час передавання даних, і найменшу затримку між сервером і кінцевими пристроями. Цей клас пристроїв споживає найбільшу кількість енергії (порівняно з класами А і В), тому зазвичай не використовує батарейного живлення.

Для створення системи з управління розумними лічильниками використовують кінцеві пристрої класу А, адже характеристики цього класу найбільше підходять для передачі даних з високою енергоефективністю.

2.3 Геоінформаційні системи

Геоінформаційні системи – це найпоширеніший вид систем для зберігання та обробки даних, що описують об’єкти у просторі. Вони широко використовуються у задачах благоустрою землі, обліку об’єктів нерухомості, систем інженерних мереж, геодезії, тощо.

2.3.1 Функції геоінформаційної системи

Геоінформаційна система це комплекс програмного, апаратного та інформаційного забезпечення, що дозволяє користувачам системи взаємодіяти та аналізувати географічні дані. Комплексність ГІС рішень обумовлена різними складовими, що мають бути представлені разом для вирішення задач.

Основними функціями будь-якої геоінформаційної системи є:

- обробка даних;
- аналіз даних;
- візуалізація даних.

Реалізація всіх функцій доступна у хмарному сервісі ArcGIS Online. Також використання такого сервісу дозволяє уникнути встановлення програми ArcGIS Desktop на персональному комп'ютері. Однією з найважливіших функцій геоінформаційної системи є управління просторовими даними [9].

В першу чергу геоінформаційні системи використовуються для зручного відображення даних.

2.3.2 Геоінформаційна база даних

Геоінформаційна база даних також є одним з найголовніших компонентів для побудови геоінформаційної системи. Геоінформаційна база даних дозволяє зберігати дані у форматі, який зручно використовувати для відображення інформації на картах [4].

Геоінформаційна база даних успадковує основні властивості, що наявні в звичайних базах даних. Основні функції, що властиві базам даним:

- цілісність даних;
- створення залежності між даними;
- висока швидкість доступу до даних;
- повний доступ до даних;
- висока надійність зв'язку;
- система управління користувачами.

Геоінформаційні СУБД представляє собою програмний комплекс, що надає можливості для управління бази даних. Основні функції СУБД:

- створення таблиць;
- додавання записів у таблиці;
- редагування записів у таблиці;
- видалення записів у таблиці;
- пошук записів у базі даних.

За допомогою СУБД, що надає функціонал з управління даними та доступу до них, розробник має можливість не замислюватись над програмною реалізацією такого функціоналу, а просто його використовувати.

2.3.3 ГІС як узагальнена інформаційна система

Геоінформаційні системи з'явились як практична потреба узагальнення інформаційних систем з просторовою локалізацією даних. Такий підхід дозволяє визначити геоінформаційні системи як ефективну автоматизовану інтегровану систему з просторовою локалізацією даних. Геоінформаційна система поєднує в собі загальні властивості інформаційних систем цього класу і є розвитком таких систем для забезпечення зручного управління просторовими даними.

Основним типом зв'язку між об'єктами геоінформаційної системи є позиціонування у системі координат поверхні, що дозволяє з'єднувати географічні координати та прості дані.

Одна з найголовніших відмінностей геоінформаційної системи від інших автоматизованих систем з просторовою локалізацією прийнято вважати використання теорії графів для створення топології лінійних і ареальних об'єктів, використання криволінійних систем координат і картографічних проекцій для зв'язку просторових об'єктів з точками земної поверхні [3].

2.4 Інженерні мережі

Інженерна мережа — основний елемент, з яким працюють користувачі при управлінні інженерними та телекомунікаційними мережами в ArcGIS. Разом із заснованою на сервісах трансакційною моделлю, правилами атрибутів, інструментами редагування і іншими можливостями вона дозволяє користувачам повністю моделювати і аналізувати комплексні мережеві системи водопостачання, газопостачання,

електропостачання, телекомунікаційні мережі, систему каналізації, стічних вод та інших елементів інженерних будівель [10].

Функціонал ArcGIS дозволяє моделювати всі компоненти системи, такі як труби, лінії електропередач, лічильники та клапани. Використовуючи функціонал програмного продукту ArcGIS можна робити моделювання, динамічний аналіз та симуляцію різних стихійних подій, на основі яких робити висновки щодо функціювання мережі у певні моменти часу.

Також інженерна мережа дозволяє моделювати те, як з'єднанні елементи інжереної системи, за допомогою функцій управління, користувач має змогу перевірити та запобігти появи неправильних зв'язків між елементами. Інженерні мережі дозволяють визначити місцезнаходження елемента за допомогою структурних ідентифікаторів, об'єднання декількох елементів у структурні об'єкти дозволяє користувачу перегляд у ефективніший спосіб.

Просторові данні, що пов'язані з інженерними об'єктами дозволяють зв'язати атрибутивні дані та географічним положенням. Таким чином, користувачі системи мають змогу переглянути у виді схеми або карти розташування об'єктів та їх атрибути. Функціонал системи дозволяє в зручному вигляді робити облік пристроїв та об'єктів, що використовуються для побудови інженерної мережі.

За допомогою засобів симуляції поведінки мережі при різних подіях, користувачі системи мають змогу отримати та аналізувати наслідки таких подій як стихійні явища, розриви між елементами мережі та інші події, що можуть негативно впливати на систему.

2.5 Аналіз існуючих аналогічних програмних систем

У зв'язку з особливостями до геоінформаційної системи ArcGIS Online, що повинна взаємодіяти із мережевим сервером Actility ThingPark Wireless, на сьогодні не

існує програмного забезпечення, яке б дозволило об’єднати та надати інтерфейс для зручної взаємодії. Особливості реалізації передачі даних від мережевого сервера, були враховані при розробці програмного продукту.

Можливість виробника лічильника шифрувати дані, своїм унікальним способом також накладає низку вимок до програмного продукту та його індивідуальність. Наявні системи, покривають тільки функціонал щодо відображення показників лічильників та їх характеристики.

Особливістю системи, що була розроблена, є інтеграція даних, які надсилає розумний лічильник у геоінформаційну систему, що представляє інженерну мережу НТУУ “КПІ”. Користувач системи має змогу передивитись у вигляді схем інженерні об’єкти з географічною прив’язкою.

Для повного аналізу та перегляду актуальної інформації щодо показників розумних лічильників, інформація надсилається з мережевого сервера обробляється та додається у правильному форматі на карту. Інтеграція з Actility ThingPark Wireless дозволяє в реальному часі дізнаватись та аналізувати показники лічильників з різних мереж.

Можливість схематичного відображення та оновлення даних в автоматичному режимі дозволяє користувачам точніше моделювати ситуації з використанням інженерних мереж, а також отримувати актуальні дані про показники споживання енергії, води та газу, що можуть бути використанні для покращення енергоефективності будівлі або всього комплексу загалом.

3. ЗАСОБИ РОЗРОБКИ СИСТЕМИ ДЛЯ ЗБОРУ, НАКОПИЧЕННЮ ТА WEB ВІДОБРАЖЕННЯ ДАНИХ ВИМІРЮВАНЬ В ІНЖЕНЕРНИХ МЕРЕЖАХ

Під час розробки системи з управління розумними лічильниками можна виділити три основні етапи створення програмного продукту:

- розробка геоінформаційної системи на платформі ArcGIS;
- налаштування мережевого сервера Actility ThingPark Wireless для роботи з лічильниками;
- розробка сервісу для передачі даних.

Для розробки Web-інтерфейсу був використаний програмний продукт ArcGIS Online, для створення сервісу передачі даних розумного лічильника – програмна платформа Node.js.

3.1 Мова програмування JavaScript

JavaScript – це динамічна, об'єктно-орієнтована мова програмування. Реалізація стандарту ECMAScript.

JavaScript класифікують як прототипну (підмножина об'єктно-орієнтованої), скриптову мову програмування з динамічною типізацією. Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування (імперативну та частково функціональну) і деякі відповідні архітектурні властивості, зокрема: динамічна та слабка типізація, автоматичне керування пам'яттю, прототипне успадкування, функції як об'єкти першого класу.

JavaScript найчастіше використовується як частина браузера, що надає можливість виконання коду на стороні клієнта та взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб-сторінки.

Мова JavaScript також використовується для програмування на стороні сервера (подібно до таких мов програмування, як Java і C#), розробки ігор, стаціонарних та мобільних додатків, сценаріїв в прикладному ПЗ, всередині PDF-документів тощо.

JavaScript, наразі, є однією з найпопулярніших мов програмування в інтернеті. Але спочатку багато професіональних програмістів скептично ставилися до мови, цільова аудиторія якої складалася з програмістів-любителів. Поява AJAX змінила ситуацію та повернула увагу професійної спільноти до мови. В результаті, були розроблені та покращені багато практик використання JavaScript, створені бібліотеки та фреймворки, поширилося використання JavaScript поза браузером. Можна зробити висновок, що мова JavaScript активно використовуються для:

- написання сценаріїв веб-сторінок для надання їм інтерактивності;
- створення односторінкових веб-застосунків (React, Angular, Vue.js);
- програмування на стороні сервера (Node.js);
- стаціонарних застосунків (Electron, NW.js);
- мобільних застосунків (React Native, Cordova);
- всередині PDF-документів тощо.

Оскільки браузери, від різних виробників, дещо відрізняються у поведінці JavaScript і реалізації Об'єктної моделі документа, необхідно мати відлагоджувач для кожного браузера, якщо веб-застосування орієнтовано на нього.

3.2 Мова програмування TypeScript

TypeScript – мова програмування, що розробляється компанією Microsoft і була представлена восени 2012 року, також позиціонується як засіб розробки веб-застосунків, що розширює можливості JavaScript.

Хоча він зародився в компанії Microsoft, і його фактичним творцем є програміст Андерс Хейлсберг, відомий як талановитий розробник таких мов як Delphi, C#, але даний проект відразу став розвиватися у середовищі OpenSource. Майже з самого початку нову мову стали швидко поширювати в силу своєї масштабованості і продуктивності. Чимало проектів, які були написані на мові JavaScript, стали переноситися на мову TypeScript.

Популярність і актуальність ідей нової мови привела до того, що ряд з цих ідей в подальшому стануть частиною нового стандарту JavaScript. А нова версія одного з найпопулярніших фреймворків для розробки веб-інтерфейсів веб – Angular повністю написана на TypeScript спільно компаніями Microsoft і Google.

TypeScript – це строго типізована і компільована мова, що дозволяє розробникам Java та C# у швидший спосіб опанувати мову для розробки веб-інтерфейсів. Хоча на виході компілятор створює JavaScript, який потім запускається і може бути оброблений браузером. Однак строга типізація зменшує кількість потенційних помилок, які могли б виникнути при розробці на мові JavaScript.

Сьогодні свою популярність мова здобула не тільки у сфері розробки веб-інтерфейсів, а й через свою популярність набула великого поширення у розробці серверних застосунків. Яскравим прикладом реалізації серверної бібліотеки є Fastify, що повністю розроблена мовою TypeScript та позбавлена типових помилок та ускладнень, які могли виникнути при розробці програмного продукту мовою JavaScript.

TypeScript реалізує багато концепцій, які властиві об'єктно-орієнтованим мовам програмування. Наприклад є реалізація такого функціоналу як успадкування, поліморфізм, інкапсуляція і модифікатори доступу і так далі.

У той же час TypeScript розширює можливості JavaScript, а це значить, що будь-яка програма на JavaSript є програмою на TypeScript. Завдяки цьому у TypeScript можна використовувати всі ті конструкції, які застосовуються в мові JavaScript, наприклад, оператори, умовні, циклічні конструкції, тощо. Більш того код написаний мовою TypeSript компілюється в JavaScript. В кінцевому рахунку, TypeSript – це всього лише інструмент, який покликаний полегшити роботу розробникам для розробки масштабованих додатків у сфері веб-інтерфейсів та серверних застосунків.

Згенерований компілятором TypeScript код JavaScript підтримується переважною більшістю браузерів, так як орієнтується насамперед на стандарт ECMAScript 3, хоча у конфігурації компілятора TypeSript також можна вказати інші стандарти ECMAScript 5 або ECMAScript 2015 / 2017. На офіційному сайті розробника TypeScript також представлена детальна інформація після ознайомлення з якою розробники, що писали на JavaScript мають можливості для переносу існуючого проекту на TypeScript.

3.3 Фреймворк NestJS

NestJS - це основа для створення ефективних масштабованих додатків на сервері Node.js. Він використовує прогресивний JavaScript, створений і повністю підтримує TypeScript.

В останні роки, завдяки Node.js, JavaScript став популярним в Інтернеті як для розробки веб інтерфейсу, так і для додатків та серверів. Це породило дивовижні проекти, такі як Angular, React та Vue, які покращують продуктивність розробників та дають змогу створювати швидкі, надійні та розширювані додатки для розробки інтерфейсу. Однак, хоча для Node.js існує безліч чудових бібліотек, помічників та інструментів,

жоден з них ефективно не вирішує головну проблему – архітектура та складність до розширення програмного коду.

Фреймворк поєднує такі концепції програмування:

- об'єктно-орієнтоване програмування;
- функціональне програмування;
- функціональне реактивне програмування.

За замовчуванням Nest використовує провірений часом HTTP-сервер Express, за бажанням, можна налаштувати також використовувати Fastify, що написаний на TypeScript.

Nest забезпечує рівень абстракції над серверною частою Node.js (Express або Fastify), але також надає доступ до API безпосередньо розробнику. Це дозволяє розробникам використовувати безліч сторонніх модулів, доступних для базової платформи, що дозволяють розширити та додати новий функціонал для додатку.

3.4 Реалізація комунікації між компонентами програмної системи

Однією з найважливіших задач системи є можливість обміну даними між собою для ефективного управління розумними лічильниками та аналізу даних, які вони передають про свої показники.

Для вирішення задач з інтеграції програмних компонентів, що повинні мати можливість взаємодіяти між собою, було виділено основні процеси для досягнення цієї цілі:

- необхідно забезпечити найшвидший спосіб комунікації систем між собою;
- необхідно забезпечити можливість інтеграції нових систем;
- необхідно розробити засоби для дешифрування даних;
- необхідно забезпечити можливість використання API, що надає функціонал для розробки геоінформаційної системи.

Найліпшим способом для комунікації всіх систем було обрано протокол HTTP.

HTTP (Hyper Text Transfer Protocol) — протокол рівня додатків для розподілених, інформаційних систем гіпермедіа. Це загальний протокол без зберігання стану між запитами, який може бути використаний для багатьох завдань поза його використанням для гіпертексту, таких як розподілені системи управління об'єктами, шляхом розширення методів запиту, кодів помилок та заголовків.

Оскільки протокол використовується для передачі показників лічильників, його доцільність у використанні на сервері для передачі даних на платформу ArcGIS Online, забезпечує простоту взаємодії між системами. Також протокол HTTP використовується не тільки для взаємодії у форматі клієнт-сервер, це дозволяє його використовувати для розробки REST сервісів.

3.5 Платформа ArcGIS Online

Найпопулярнішим розробником у сфері геоінформаційних продуктів є американська компанія ESRI, що розробляє програмне забезпечення настільних та серверних додатків. Програмне забезпечення для настільних комп’ютерів дозволяє розробникам вирішувати геоінформаційні задачі, в тому числі задачі з картографії, сбір даних та їх аналіз, управління зображеннями, а також доступ до просторової інформації.

Сьогодні хмарні технології активно застосовуються у сфері розробки програмного забезпечення. Ця технологія надає користувачам мережі Інтернет, доступ до комп'ютерних ресурсів сервера і використання програмного забезпечення як онлайн-сервісу.

Хмарні сховища дозволяють розміщувати та зберігати великі об’єми даних, на відміну від звичайних серверів та персональних комп’ютерів. Можливість збільшення обчислювальної потужності та постійного доступу до ресурсу означає, що тепер задачі

для підтримки оптимальної потужності та реагування на ситуації у випадку виходу з ладу звичайного серверу перекладаються на провайдера хмарної платформи.

Компанія ESRI не є виключенням, і також створила хмарний сервіс ArcGIS Online, як платформи для спільної роботи у розробці геоінформаційних систем. ArcGIS Online дозволяє користувачам у всьому світі створювати, редагувати та ділитися картами, сервісами та іншими ресурсами геоінформаційних систем.

ArcGIS Online містить велику різноманітність ресурсів, включаючи веб-карти, що можуть складатися з декількох шарів, сервісів для обробки даних, а також сервіси для збереження даних у різних форматах. ArcGIS Online також забезпечує обмін даними та їх повторне використання, таким чином дозволяючи інтегрувати існуючі рішення на свої карти. Для доступу до ресурсів, що знаходяться у хмарному середовищі також використовують Web API, що дозволяє робити операції з видалення, редагування та додавання нових ресурсів на платформу.

ArcGIS Online надає функціонал для можливості адміністрування бази даних, таким чином можна визначити користувача або групу користувачів та надати їм особливі властивості та права для взаємодії з даними.

3.6 Симуляція вхідних даних

Найчастіше для роботи з API використовується Postman – застосунок, який надає графічний інтерфейс для роботи з HTTP протоколом, дозволяє експортувати та імпортувати запити в колекції, додавати опис та коментарі до кожного запиту та навіть писати тести за допомогою JavaScript [11].

Postman вперше здобув популярність у 2012 році і став надійною та нативною для найпопулярніших операційних систем програмою, яку зараз використовують понад 8 мільйонів розробників та 300 000 компаній. Інші програми мають доволі обмежений набір функціональних можливостей у порівнянні з Postman [12].

На (рисунку 3.1) зображено користувацький інтерфейс програми, що є одним з найголовніших переваг Postman. Також Postman надає додаткові функціональні можливості на кшталт шаблонізації, що дозволяє легко перемикатись між локальним та розробницьким середовищем. Також до таких можливостей можна віднести швидкий пошук, автоматичне форматування для різних форматів даних, збереження історії.

Для розробки програмного забезпечення, що взаємодіє по протоколу HTTP найліпшим вибором є Postman. Користувач має змогу створити колекції з запитів до сервісу, що дозволяє розробляти систему та покривати її інтеграційними тестами. Можливість створення різних змінних, що можуть змінювати своє значення в ході роботи програми та реагувати на зміну середовища, дозволяють автоматизувати процес збору необхідних змінних та їх зберігання. Програмний продукт надає функції для імпорту сторонніх колекцій та експорту власної колекції для швидкого налаштування системи та можливості перегляду створених колекцій іншими розробниками.

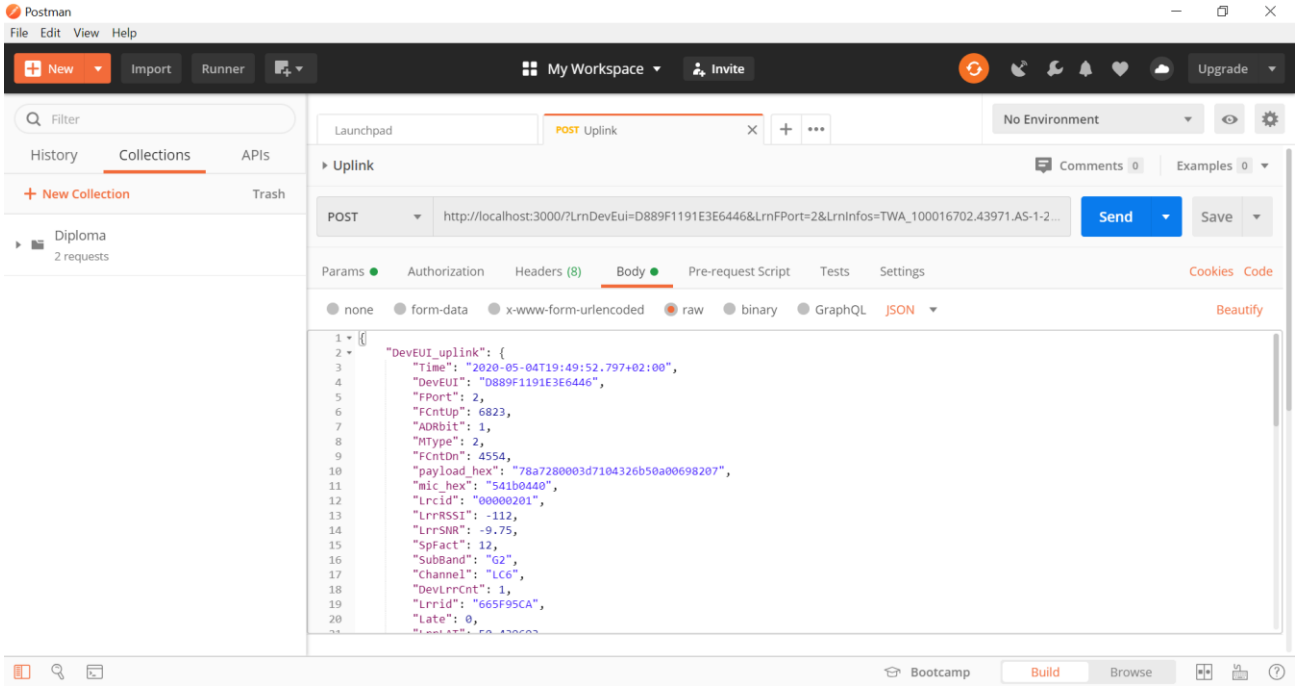

Рисунок 3.1 – Користувацький інтерфейс Postman

Можливість створення тестових запитів дозволяє розробнику розробляти програмний продукт та не створювати додаткових запитів до реальних сервісів, що можуть використовуватись.

Завдяки функції створення копій запитів, користувач має змогу модифікувати та налаштовувати відповідним чином запити, що будуть відправлені на адресу, яка вказана.

3.7 Середовище розробки JetBrains Webstorm

Програмне забезпечення JetBrains WebStorm являє собою інструмент для розробки web-сайтів і редагування HTML, CSS і JavaScript коду. Рішення забезпечує швидку навігацію по файлах і генерує повідомлення про виникаючі проблеми в коді в режимі реального часу. JetBrains WebStorm дозволяє додавати розмітку HTML-документів або елементів безпосередньо в JavaScript. JetBrains WebStorm здійснює розгортання і синхронізацію проектів через протокол FTP.

Використовуючи можливості коду HTML і XML, WebStorm забезпечує автоматичне завершення стилів, посилань, атрибутів і інших елементів коду. При роботі з CSS здійснюється завершення коду класів, HTML-номерів, ключових слів. WebStorm пропонує автоматичне рішення таких проблем, як вибір формату, властивостей, класів, посилань на файли і інших атрибутів CSS. Рішення дозволяє використовувати потужність інструменту Zen coding для верстки HTML, відображає дії тега на web-сторінці.

Продукт WebStorm здійснює завершення коду JavaScript для ключових слів, лейблів, змінних, параметрів і функцій DOM і підтримує специфічні особливості популярних браузерів. Реалізовані в рішенні функції рефакторінгу JavaScript дозволяють перетворювати структуру коду і файлів.

WebStorm забезпечує налагодження коду JavaScript і надає широкий діапазон можливостей: знаходження помилок в JavaScript, налаштування параметрів точки

зупинники під час відлагодженя, тестування синтаксису коду в режимі реального часу. Продукт підтримує платформу Node.js, ReactNative та інші сучасні фреймворки.

3.8 Система контролю версій Git

Системи контролю версій дозволяють розробникам зберігати всі зміни, внесені в код та файли, які користувач вказав для відслідковування. Тому у випадках, коли через певний час є проблеми з компіляцією коду або інші помилки система контролю версій допомагає відновити файли та код до їх працюючого стану, що дозволяє уникнути ситуацій з витраченням великого проміжку часу для вирішення проблем.

Системи контролю версій також дають можливість декільком розробникам працювати над одним проектом і зберігати внесені зміни, та мати єдиний спосіб дистрибуції програмного коду та файлів проекту.

Існує три основних типи систем контролю версій: локальна, централізована і розподілена.

Локальний тип використовуються та широко застосовуються у випадках збереження історії файлів та їх копії у відповідну директорію, що являє собою локальну базу даних з версіями файлів.

Централізовані системи були створені для вирішення проблеми взаємодії з іншими розробниками та користувачами. Такі системи мають єдиний сервер, який містить всі версії файлів. Користувачі, можуть отримувати файли з цього централізованого сховища. Проте, такий підхід має істотний недолік – вихід сервера з ладу спровокує втрату всіх даних, що зберігались на сервері.

Недолік централізованих систем був виправлений в розподілених системах, користувачі яких не просто скачують файл певної версії, а повністю копіюють базу даних з версіями файлів. Це означає, що у кожного клієнта є копія всього вихідного проекту і внесених змін до нього. В цьому випадку, якщо один з серверів вийде з ладу,

будь-який інший користувач системи може завантажити свій проект на сервер для продовження роботи з іншими користувачами. Також перевагою розподілених систем контролю версій є можливість роботи з декількома серверами віддалених проектів, що можуть бути синхронізовані між собою.

Git – це розподілена система контролю версій, яка дає можливість розробникам відстежувати зміни в файлах і працювати спільно з іншими розробниками. Вона була розроблена в 2005 році Лінусом Торвальдом, який являється творцем Linux, для того, щоб інші розробники могли вносити свої покращення та модифікації в ядро Linux. Git відомий своєю швидкістю, простим дизайном, підтримкою нелінійної розробки, повної децентралізацією і можливістю ефективно працювати з великими проектами.

Git має віддалене сховище, яке зберігається на сервері, і локальне сховище, яке зберігається в комп'ютері кожного розробника. Це означає, що файли не просто зберігається на центральному сервері, але повна копія проекту також присутня на всіх комп’ютерах розробників (рисунок 3.2).

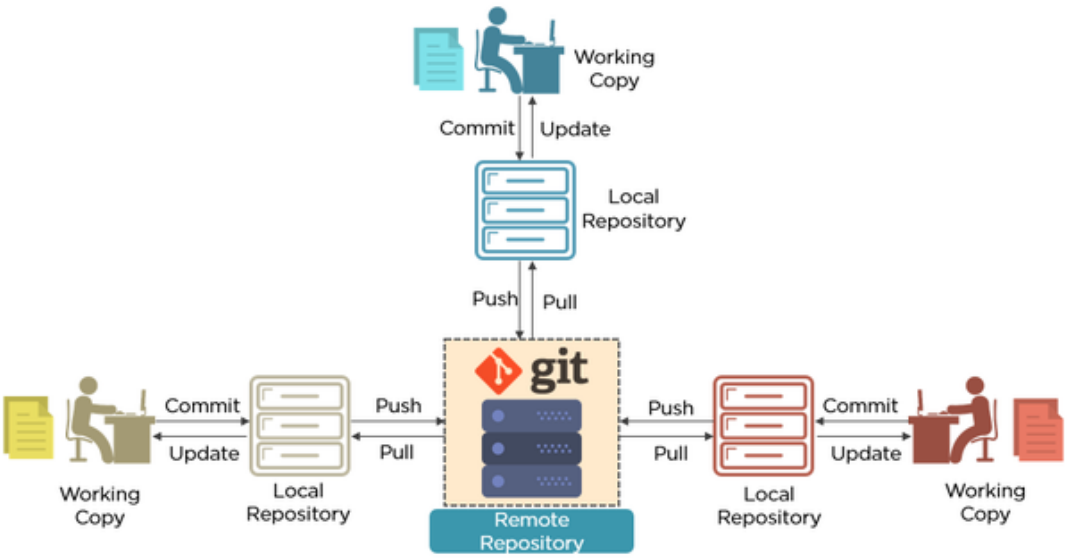

Рисунок 3.2 – Схема роботи користувачів з Git

Git варто відокремити від інших систем контролю версій через підхід до роботи з великими даними. Більшість інших систем контролю версій зберігають інформацію у вигляді списку змін в файлах. Замість цього, підхід який застосовуються в Git для

зберігання даних більше схожий на набір знімків мініатюрної файлової системи. Кожен раз, коли ви зберігаєте файли свого проекту в Git, система запам'ятовує, як виглядає кожен файл в цей момент, і зберігає посилання на цей знімок.

Найголовнішими перевагами Git відносно інших систем контролю версій є:

- безкоштовний для використання;
- швидкість роботи;
- невеликий розмір;
- резервне копіювання;
- зручне управління гілками;
- зручний командний інтерфейс;

Одним з найпопулярнішим хостингів для Git є сервіс GitHub. Хостинговий сервіс Git – це онлайн-база даних, яка дозволяє відслідковувати та ділитися своїми проектами управління версіями Git за межами локального комп'ютера або сервера. На відміну від Git, GitHub працює виключно на хмарній основі. GitHub також розширює основні функціональні можливості Git. Він представляє надзвичайно інтуїтивний, графічно представлений інтерфейс користувача та надає програмістам вбудовані засоби управління та управління задачами. Додаткові функції можна реалізувати за допомогою сервісу GitHub Marketplace. Оскільки GitHub базується на хмарі, до окремих сховищ Git може бути віддалений доступ будь-якою авторизованою особою, з будь-якого комп’ютера та в будь-якій точці світу.

За допомогою GitHub ви можете ділитися своїм проектом та файлами з іншими, надаючи їм можливість редагувати та вносити зміни у ваш Git проект. Також доступний настільний додаток GitHub, який пропонує деякі додаткові функції для досвідчених розробників.

Існують також інші онлайн хостинги, що надають послуги сховища Git: GitLab, BitBucket та SourceForge найпопулярніші альтернативи GitHub. GitLab має функцію, що дозволяє користувачам GitHub мігрувати свої проекти безпосередньо в GitLab.

4. ОПИС РЕАЛІЗАЦІЇ СИСТЕМИ ДЛЯ ЗБОРУ, НАКОПИЧЕННЮ ТА WEB ВІДОБРАЖЕННЯ ДАНИХ ВИМІРЮВАНЬ В ІНЖЕНЕРНИХ МЕРЕЖАХ

Реалізація геоінформаційної системи для управління розумними лічильниками передбачає налаштування систем, що надають можливість зі зручного встановлення розумних лічильників та їх налаштування. Також система повинна забезпечувати надійний зв’язок для передачі даних та можливість масштабування у випадку збільшення кількості лічильників або підключення інших розумних датчиків.

Також реалізація включає розробку геоінформаційної бази даних, де будуть зберігатись данні, які були надісланні розумним лічильником у правильному форматі для можливості експорту даних з таблиці та використання цієї таблиці для розробки нових ArcGIS застосунків.

Ключовим елементом для об’єднання системи контролю та управління розумними лічильниками, які використовують програмний продукт компаній Actility та OrionM2M у поєднанні з Web-інтерфейсом, що надає можливість користувачу взаємодіяти з даними є розробка серверного застосунку. Серверний застосунок у поєднанні з ArcGIS Online дозволяє переглядати інженерні мережі, які створенні за допомогою програмного продукту ArcGIS, експортувати накопичені дані у геоінформаційній базі даних, а також робити аналіз цих даних.

4.1 Мережевий сервер Actility

Для зв’язку розумних датчиків, шлюзів та програмних продуктів використовується мережевий сервер Actility, що забезпечує надійну та безпечну маршрутизацію даних по мережі LoRaWAN. Одночасно з системою підтримки операцій

(Operation Support System) – це дозволяє розробити програмний продукт у моделі програмне забезпечення як послуга. Використання цього сервісу дозволяє користувачам використовувати надані інтерфейси у своїх цілях, при цьому не замислюватись над деталями та способами реалізації цього функціоналу на сервері.

Ключовою особливістю додатків, що розробляються з використанням серверу Actility, звичайно є легкість підключення нових пристроїв та їх налаштування. Технологія відкритого стандарту LoRaWAN є перевіреним лідером у застосуванні розумних лічильників, що постачає рішення, які легко масштабувати, оптимізуючи експлуатаційні витрати та зменшуючи комунальні відходи. Платформа підключення ThingPark IoT від Actility добре обладнана для того, щоб стати лідером на цьому швидко зростаючому ринку рішень розумних пристроїв на основі IoT, керуючи як приватними, так і громадськими мережами та підтримуючи екосистему датчиків і програм IoT.

На (рисунку 4.1) зображено зв’язок між елементами системи та порядок передачі даних, що генерують кінцеві пристрої LoRaWAN мережі, до яких можна віднести розумні датчики та лічильники. Схема роботи сервісу дозволяє користувачам розробляти власні програмні рішення за рахунок інтерфейсу користувача, який має змогу робити налаштування кінцевих пристроїв.

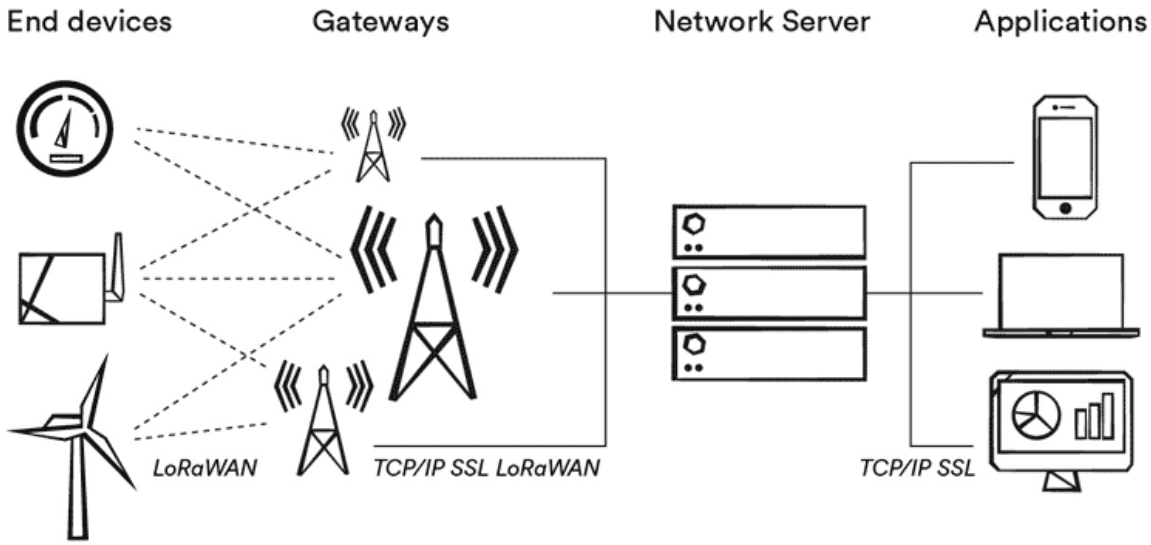

Рисунок 4.1 – Архітектура сервісу Actility

Actility розгортає єдину, масштабовану, багатоцільову мережеву інфраструктуру IoT для комунальних підприємств та міст, забезпечуючи чітко визначені точки взаємодії між системами та значно спрощує розробку систем зменшуючи інтеграційні витрати та труднощі, що виникають внаслідок створення різновидів лічильників. Тож компанії, що забезпечують електропостачання, газопостачання та водопостачання міста можуть зосередитись на створенні якісного продукту завдяки сервісу, що надає послуги з'єднання лічильників у єдину систему.

За рахунок своєї бізнес-моделі користувачі системи мають змогу масштабувати свої рішення та використовувати лиш необхідний функціонал для створення нової системи.

4.2 Радіо-модем OrionMeter для передачі показників

На території НТУУ "КПІ" для передачі показників встановлені радіо-модеми фірми OrionM2M, що дозволяють перетворити звичайні лічильники на розумні, основними перевагами, використання продукції цієї компанії є:

- термін роботи без зміни батареї від 7 років;
- дистанцій сбір показників;
- легкість монтажу та обслуговування;
- змінна батарея;
- захист від пошкоджень та втручання.

Також фірма OrionM2M надає користувачам інтерфейс для можливості автоматизації збору показників з лічильників, моніторингу в режимі реального часу параметрів використання ресурсів, сбір статистики та прогнозування.

Програмний продукт OrionAMR дозволяє користувачам у веб-інтерфейсі подивитись всі показники системи, що використовує датчики або розумні лічильники OrionM2M.

Основні особливості OrionAMR:

- перегляд статистики по кожному пристрою за необхідний період часу;
- автоматична передача показників в облікову систему;
- перегляд динаміки споживання ресурсів;
- експорт даних на різні платформи;
- доступ до даних різними користувачами системи.

Можливість зручного аналізу та перегляд статистики в реальному часі дозволяє користувачам системи OrionAMR реагувати на критичні показники або їх відхилення від норми. Таким чином система надає функціонал використання якого допомагає знизити витрати на сервісні служби та оптимізувати дії, які персонал повинен виконати для зняття достовірних показників і їх обробку. Аналіз даних, що отримані по кожному з датчиків дозволяє формувати звіти та робити висновки щодо енергоефективності системи та її подальшого покращення.

Питання енергоефективності системи є одним з найважливіших показників на основі цього аналітики можуть побудувати моделі та сформувати звіти для вдосконалення будівель, приміщень, тощо.

У зв’язку з малою собівартістю радіо-модулів, використання яких робить процес вдосконалення існуючих лічильників легшим та простішим для налаштування.

Автономний радіо-модем Orion Meter є пристроєм класу А за класифікацією LoRaWAN. Пристрій призначений для бездротового зчитування показань за допомогою розрахунку числа обертів оптичного диска лічильника з лічильників гарячої або холодної води з періодичним накопиченням значень лічильників в незалежній пам'яті пристрою з подальшою передачею накопичених даних, а також передачі даних з поточними значеннями, за допомогою радіозв'язку в мережу LoRaWAN.

4.3 Структура програмного забезпечення

Для реалізації задачі передачі та управління даними лічильників в інженерних мережах був розроблений серверний застосунок, який може бути розгорнутий на сервері з Windows або Unix платформою. Основною вимогою для програмного продукту є швидкість та легкість у масштабуванні в процесі розробки.

Популярність платформи Node.js та TypeScript надає можливості для створення високоефективних та надійних рішень на їх основі. Програмний продукт було реалізовано з використанням фреймворку Nest.js на платформі Node.js, що означає високу швидкість розробки нових компонентів системи та легкість в освоєнні та розумінні роботи системи в цілому. Використання фреймворку Nest.js дозволяє організувати архітектуру проекту, що має впорядковану структуру директорій та модулів (рисунок 4.2).

Структура розробленого проекту складається з основних директорій: dist, node_modules, src, test та файлів .gitignore, package.json, tsconfig.ts, .eslintrc.js, nest-cli.json.

Директорія src містить логіку програмного продукту та файл з налаштуваннями для запуску проекту на певному середовищі, що дозволяє змінювати значення змінних відповідно до середовища з власними налаштуваннями. Піддиректорії формуються за реалізацію різного функціоналу та наявністю функціональних тестів цих модулів, кожен з модулів реалізує певний інтерфейс, що дозволяє обробляти дані, які зв’язані з цим модулем.

Директорія dist містить єдиний файл, що містить в собі всю логіку застосунку, але код цього файлу мініфікований та оптимізований, для швидкої роботи на сервері.

Директорія node_modules містить залежності для проекту та його пакетів, що можуть бути використанні в процесі написання програмного коду. Ця директорія

формуються автоматично при створення програмного коду на платформі Node.js, яка використовує пакетний менеджер npm.

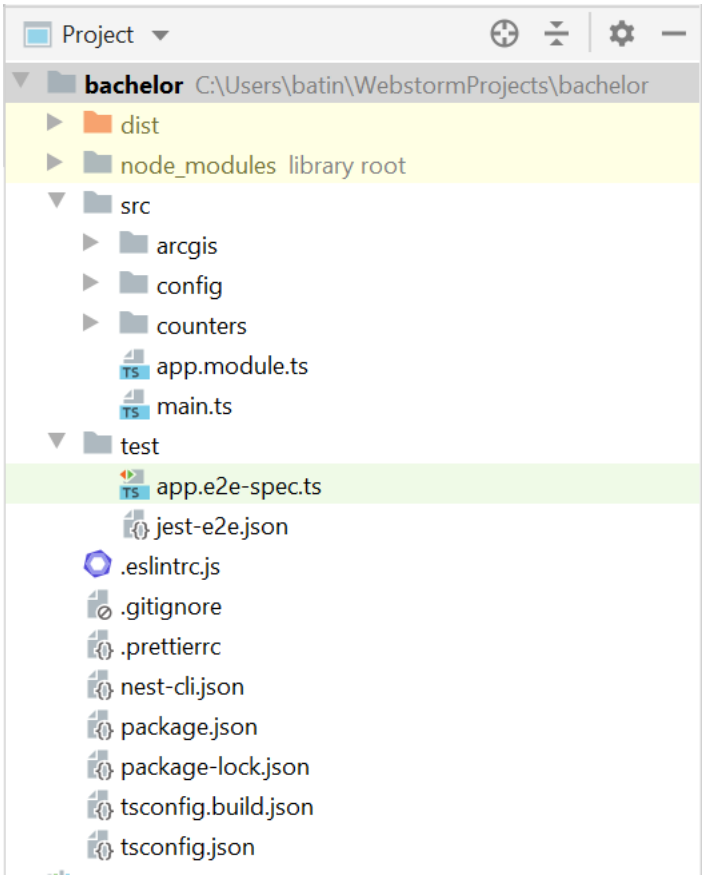

Рисунок 4.2 – Структура файлів проекту

Директорія test містить файли, які покривають систему загалом і дозволяють перевірити роботи системи в цілому. У випадках появи нових модулів, додаються нові сценарії для перевірки роботи системи в цілому.

Файл .gitignore додається для системи контролю версій. Він слугує допоміжним механізмом підтримки системи контролю версій, якщо програміст вважає, що стан деяких файлів проекту впродовж етапу розробки не змінюється, або зміни цих файлів не впливають на кінцеву роботу програми можуть бути додані у цей файл.

Файл package.json містить опис пакетів, які потрібно встановити у зв’язку з наявністю сторонніх бібліотек, що повинні бути встановлені в проекті. Також у цьому файлі є можливість описати автора, коротку інформацію та версію проекту, який

розробляється. Всі залежності, які розробник вказав в цьому файлу будуть інстальовані в директорію node_modules.

Файл tsconfig.ts дозволяє налаштовувати та описувати конфігурацію для компілятора TypeScript, вказавши версію в яку буде скомпільовані файли, шлях до файлів або регулярний вираз для пошуку файлів.

Файл eslintrc.js використовуються на проекті для задання єдиного стилю коду та його основних конструкцій. Таким чином у випадках коли програмний продукт підтримується та розробляється декількома розробниками – це дозволяє уникнути ситуації, з різним форматуванням та стилем коду.

Файл nest-cli.json генерується при використанні фреймворку Nest.js автоматично, цей файл забезпечує та надає інтерфейс для командного рядку з командами, які поліпшують та прискорюють написання типових модулів.

Як вже було зазначено вище, модуль src містить файли, які описують бізнес-логіку продукту. Директорії цього модуля поділені за принципом своєї відповідальності. Таким чином це дозволяє уникнути ситуацію з великим списком файлів і їх неструктурованістю. Кожен модуль містить файл з тестами, що забезпечує надійність у правильній роботі програмного коду. У директорії config містяться файли, які являються конфігурацією для проекту. В основі цих файлів лежить визначення змінних, що можуть бути змінені та налаштовані відповідно до середовища запуску програми.

Відповідно до архітектури фреймворку Nest.js серверна частина коду поділяються на такі частини: module, controller та provider. Файли з розширенням .controller.ts позначають частину коду, що описує відповідний функціонал для обробки вхідних запитів. За допомогою декораторів, Nest.js описує класи та розуміє, як компоненти взаємодіють між собою. За допомогою файлів з розширенням .module.ts, ми маємо змогу виділити набір компонентів, що відносяться до одного модуля та їх повторне використання в інших місцях за потреби. Також фреймворк має низку різних декораторів, що дозволяють реалізувати функціонал для покриття потреб розробника і

їх розширити у випадках, якщо такий функціонал не існує або розробник потребує більшого контролю ситуації при розробці програмного продукту.

4.4 Створення сервісу для передачі даних до ArcGIS Online

Основною задачею сервісу є забезпечення функції обробки вхідних пакетів, що надходять від розумних лічильників та їх обробку і трансформацію для коректної роботи з ArcGIS Online REST API, що надає функціонал для додавання, редагування та управління інформацією у хмарному сервісі. Особливістю вхідних даних від розумних лічильників є закодована інформація, яку потрібно розшифрувати. Дані, що надсилає розумний лічильник закодовані у форматі ASCII, для розшифрування інформації було створено програмний модуль, що виконує необхідні дії для розшифрування поля з даними, які відправляє розумний лічильник.

На (рисунку 4.3) зображено діаграму послідовності, що показує обробку вхідних пакетів, які надсилає датчик. Діаграма приховує особливості реалізації функціоналу, але надає можливість сформувати та надати базове розуміння роботи системи, що функціонує для вирішення поставленої задачі з передачі даних до ArcGIS Online.

ArcGIS Online надає розробникам інтерфейси для взаємодії з різними сервісами та модулями цієї системи. Для розробки надійного та захищеного каналу для передачі даних, всі данні відправляються з токеном, що надсилає до системи інформацію про користувача. Методи автентифікації у програмних продуктах ArcGIS Online дозволяють створювати кросплатформені рішення використовуючи єдиний метод авторизації.

Для роботи з таблицями та шарами використовуються методи ArcGIS Online, що дозволяють розробникам вносити, редагувати та видаляти інформацію. Наявність документації, що показує правильний формат запиту до сервісу дозволяє уникнути проблем в процесі розробки застосунку.

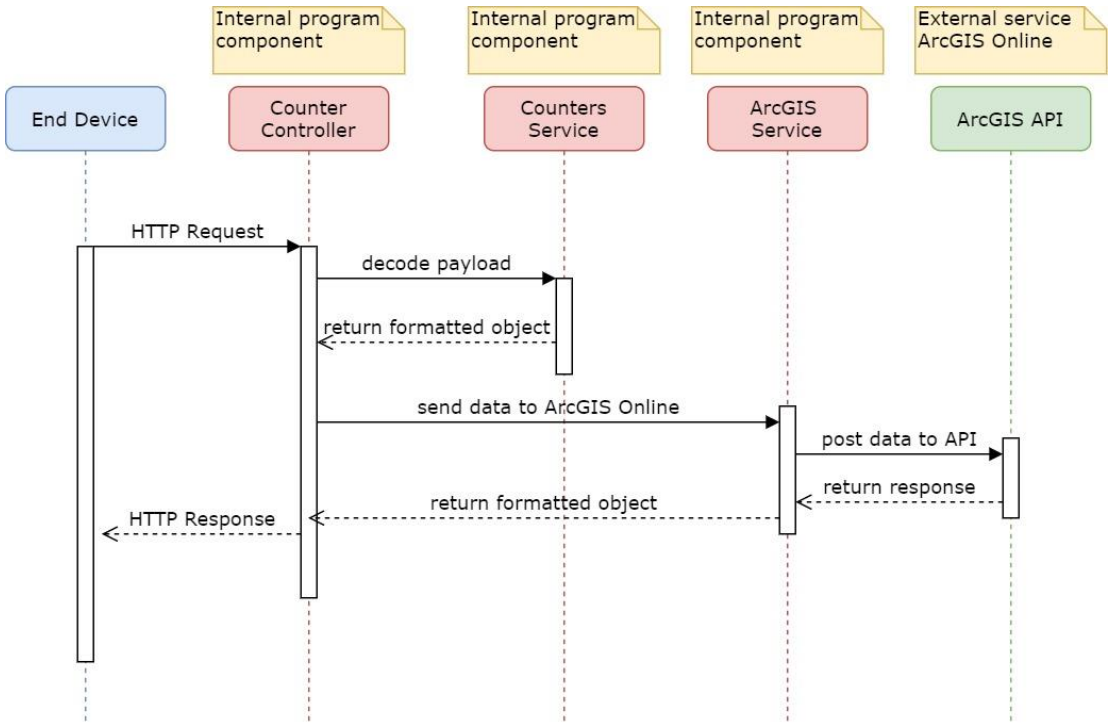

Рисунок 4.3 – Діаграма послідовності

Для забезпечення коректної роботи програмного коду також були описані функціональні тести Отже, серверна частина програмного продукту реалізована з використанням фреймворку Nest.js надає функціонал для обробки вхідних пакетів даних, що надсилаються з розумних лічильників та їх обробку для використання в ArcGIS Online, який надає користувачам зрозумілий інтерфейс взаємодії з даними, що зберігаються в системі.

Інтеграція з багатьма базами даними та платформами дозволяє майбутнім розробникам реалізовувати та додати нові функції, що розширять функціонал системи. Підтримка типів та ідеологія фреймворку допомагає розробникам без великого досвіду масштабувати та робити рефакторінг з метою покращення системи [13].

4.5 Створення Web-додатку на платформі ArcGIS Online

Для відображення інженерних мереж, а також даних про розумні лічильники та їх показники було вирішено створити Web-додаток на платформі ArcGIS Online, що дозволяє створювати геоінфомаційні системи та бази даних до таких систем.

Переваги хмарних рішень у простоті налаштування та зручності користування. Можливість редагування та розробка карти у веб-середовищі позбавляє користувачів та розробників до налаштування та встановлення програмного забезпечення на стаціонарному комп’ютері, однак необхідний доступ до мережі Інтернет.

Можливість створення різних шарів та геоінформаційної бази даних робить цей сервіс також простим для користувачів. Можливість зручного перегляду зав’язків між таблицями та простота фільтрації шарів, що можуть відображатися на карті також робить цей сервіс популярним для використання у сфері розробці геоінформаційних систем.

Усі таблиці, що містять інформацію про просторові об’єкти включаються у себе інформацію про відображення елемента на карті та його основні атрибути, які користувач може змінювати прямо на карті.

Для передачі даних із сервера було використано ArcGIS Online REST API, що дозволяє методом HTTP протоколу взаємодіяти з інформаціє, що знаходиться на сервері.

5. РОБОТА КОРИСТУВАЧА З СИСТЕМОЮ

Розроблений програмний продукт має дві складові – серверний застосунок та веб-застосунок ArcGIS Online, тому вимоги до серверної частини та веб-інтерфейсу відрізняються і будуть описані нижче. Вимоги до сервера включають вимоги не тільки в програмному забезпеченні, а й в апаратному. Оскільки серверна частина є обов'язковою частиною системи, що надає функції для передачі даних, вона повинна бути завантажена на сервер та бути увімкненою для коректної роботи системи. Що стосується веб-інтерфейсу, то користувач може використовувати даний застосунок просто зайшовши на посилання або у випадках необхідності може імпортувати програмний продукт та відкрити його у програмному забезпеченні ArcGIS Desktop для роботи на стаціонарному комп'ютері без з'єднання з мережею Інтернет.

Основною перевагою використання платформи Node.js є можливість встановлення на великій кількості операційних систем та легкість налаштування сервера для роботи програмного продукту.

5.1 Системні вимоги

Для роботи коректної роботи користувача з розробленою програмною системою користувачу потрібні мінімальні потужності апаратного забезпечення (таблиця 5.1).

Таблиця 5.1. Вимоги до апаратного забезпечення для перегляду веб-інтерфейсу

Пристрій	Характеристика
Процесор	Intel ® Core ™ 2 / 2 Duo/ Pentium® / Celeron ® / Xeon™ / i3 / i5 / i7 чи AMD 6 / Turion ™ / Athlon ™ /

	Duron ™ / Sempron ™ з тактовою частотою не нижче 1.5 GH
Оперативна пам’ять (RAM − Random Access Memory)	Рекомендовано не менше 1 GB RAM
Швидкість з’єднання з інтернет	Рекомендовано не менше 128 кб/сек

Апаратні архітектури, що підтримує програмне забезпечення:

- 32-розрядна (x86);
- 64-розрядна (x64).

Для ефективної роботи серверного застосунку є також певні вимоги до апаратного забезпечення на сервері, на якому буде розгорнута мережа для запуску програми у середовищі (таблиця 5.2).

Таблиця 5.2. Вимоги до апаратного забезпечення сервера

Пристрій	Характеристика
Процесор	Intel ® Core ™ / i3 / i5 / i7 чи AMD K5 ™ / K6 ™/ K7 ™ з тактовою частотою не нижче 3.0 GHz.
Оперативна пам’ять (RAM − Random Access Memory)	Рекомендовано не менше 4 GB RAM
Швидкість з’єднання з інтернет	Рекомендовано не менше 1024 кб/сек
Вільне місце на жорсткому диску	Рекомендовано не менше 50 GB

Також для можливості роботи програмного продукту на сервері, потрібно встановити платформу на якій програма може бути запущена та налаштовано відповідно до вимог користувача (таблиця 5.3).

Таблиця 5.3. Вимоги до програмного забезпечення сервера

Розділ	Назва
Операційна система	Windows Vista, 7, 8, 8.1, 10, Mac OS 10.10, 10.11, 10.12, Linux
Встановлені програми	Node.js 10 версії та вище
Встановлені компоненти	NPM

Вимоги до апаратного забезпечення є рекомендаційними та можуть бути змінені або доповнені відповідно до вимог системи, для забезпечення надійної на коректної роботи програмного забезпечення.

5.1 Перегляд інженерних мереж у Web-інтерфейсі

Web-інтерфейс надає користувача системи можливість перегляду інженерних мереж на території НТУУ “КПІ”. На (рисунку 5.1) зображено інтерфейс користувача, що має основні елементи для зручної взаємодії з картою. Можливість зміни масштабу карти дозволяє користувачам системи налаштувати вигляд карти зважаючи на роздільну здатність дисплею.

При натисканні користувача на елемент карти, буде відкрито діалогове вікно з інформацію та мінімальним описом елемента (рисунок 5.2). Основні функції, що доступні у нижній частині діалогового вікна, такі як зміна масштабу до максимального для відображення конкретного елемента, редагування інформації та прокладання маршруту до об’єкта, дозволяють користувачу у мінімальну кількість дій отримати необхідні дані.

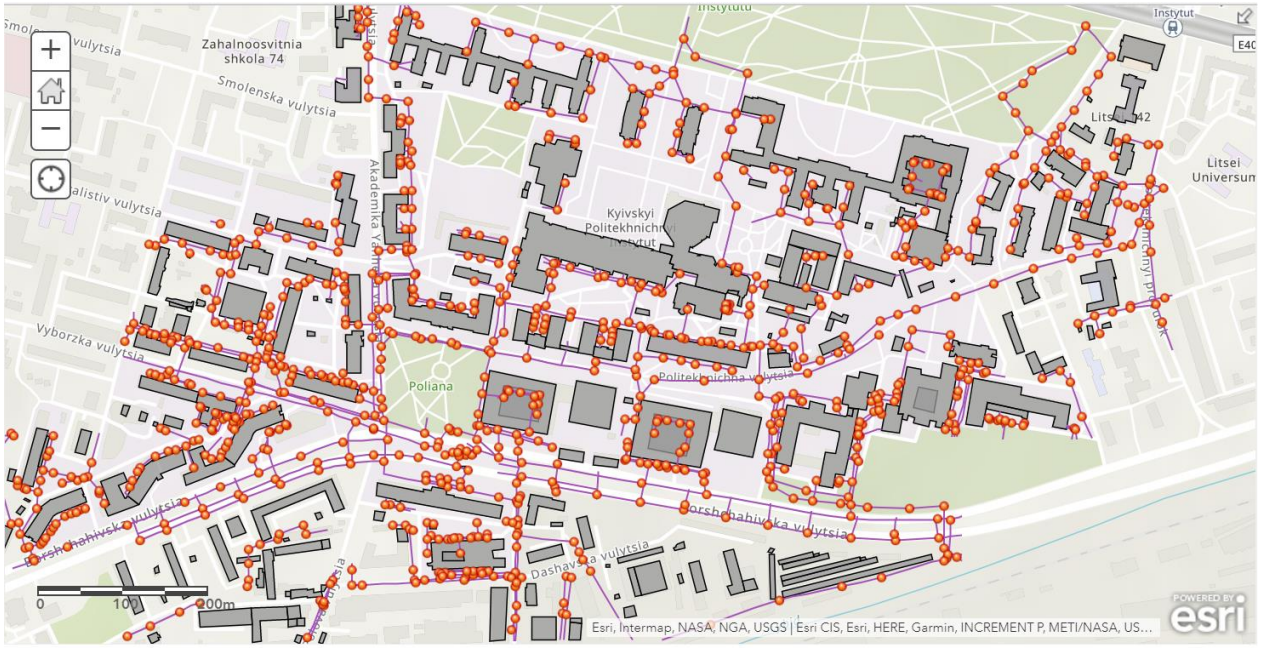

Рисунок 5.1 – Відображення інженерних мереж у веб-додатку

Для перегляду даних, що відносяться до конкретного об’єкту є посилання “Show Related Records”, що дозволяє відобразити записи, що зв’язані з об’єктом.

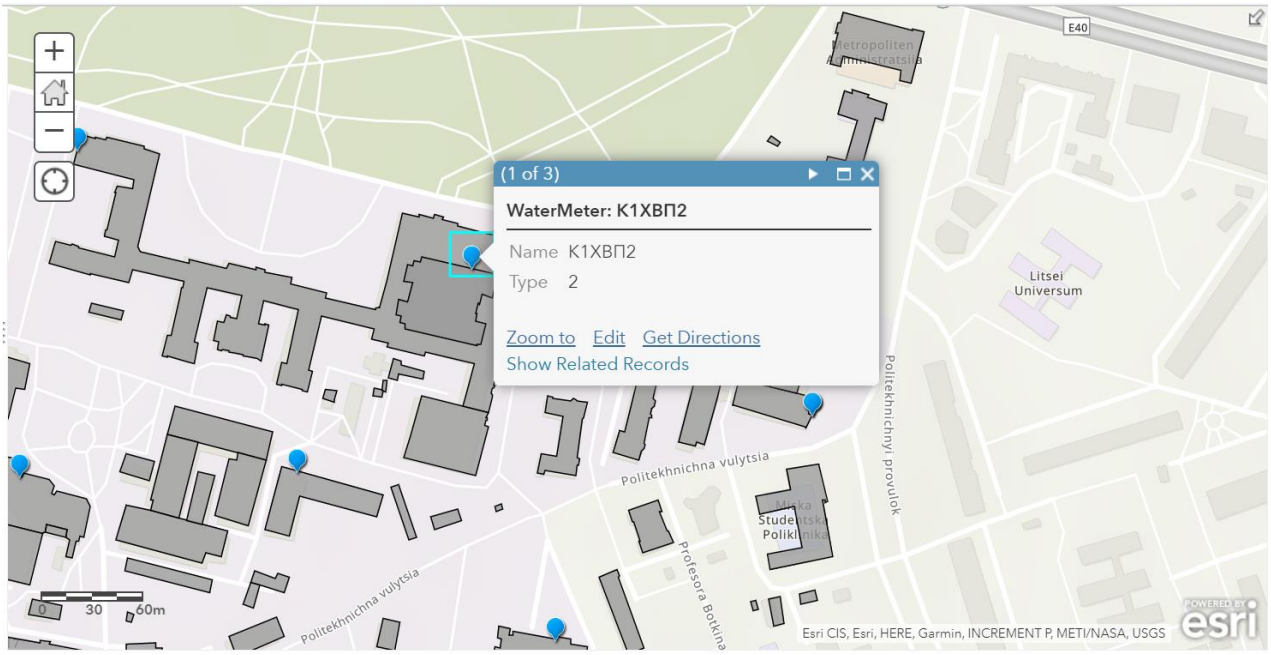

Рисунок 5.2 – Діалогове вікно об’єкту

Для налаштування відображення декількох мереж користувач має змогу фільтрації та вибору шарів карти, що відбувається за рахунок проставляння прапорців, які знаходяться перед назвою шару та його коротким описом (Рисунок 5.3).

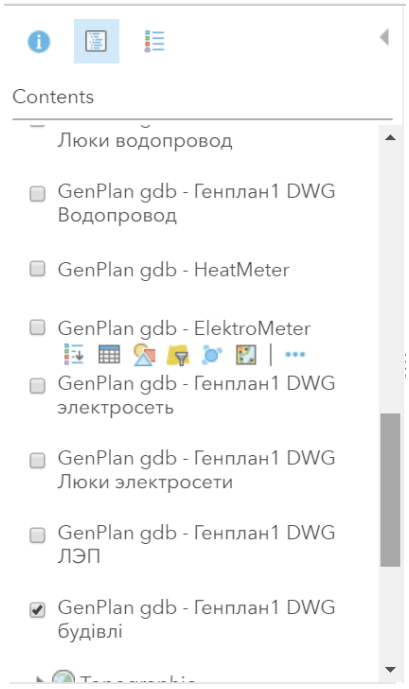

Рисунок 5.3 – Таблиця контенту ArcGIS Online

Також у віджеті, що відображає таблицю контенту, можна побачити при наводженні миші контекстне меню, що дозволяє показати структуру таблиці у якій дані зберігаються, змінити стиль відображення об’єктів та виконати аналіз даних.

5.2 Перегляд таблиць геоінформаційної бази даних

Web-інтерфейс надає користувачам системи можливість перегляду таблиць геоінформаційної бази даних, що зберігається на серверах ArcGIS Online. На (рисунку 5.4) зображено інтерфейс користувача, що має основні елементи для зручної взаємодії з таблицями, що надає доступ до основних процесів взаємодії користувача з геоінформаційною базою даних.

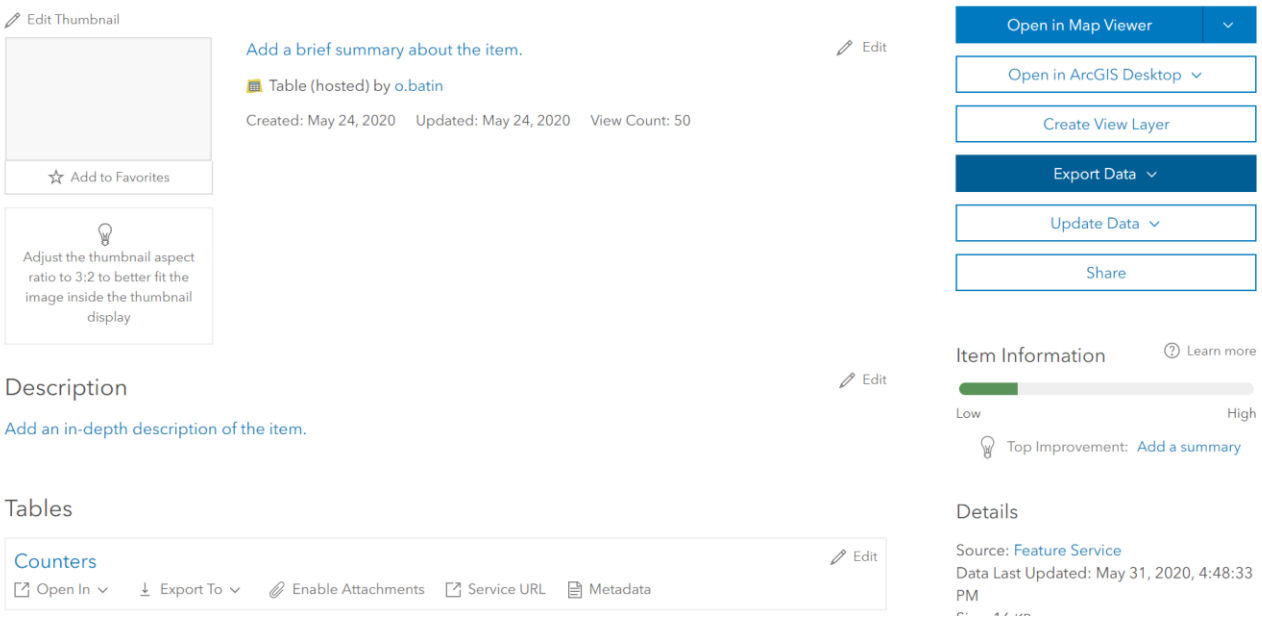

Рисунок 5.4 – Інтерфейс для взаємодії з геоінформаційною базою даних

Інтерфейс також надає доступ до експорту даних та оновлення даних з файлів, що мають розширення csv, xls, geojson. Можливість для редагування основних атрибутів таблиць та копіювати посилання на сервіс, що зберігає дані по кожній з таблиць. Користувач має змогу також редагувати опис і загальну інформацію до бази даних, що дозволяє стороннім користувачам зрозуміти, які дані містить геоінформаційна база даних. Також важливим фактором для використання онлайн сервісу є можливість надання прав іншим користувачам, що також можуть мати доступ до ресурсу. Налаштування параметрів доступу дозволяє визначати права користувачів системи, що можуть бути використанні для редагування, додавання та видалення інформації. Власник сервісу також має доступ до редагування атрибутів таблиць та їх параметрів, наприклад, тип даних атрибуту, назва та ліміти.

У вкладці “Usage” користувач має можливість перегляду кількості запитів до таблиці за кожний день та загальну характеристику з середньої кількісті запитів в день (рисунок 5.5).

Рисунок 5.5 – Діаграма запитів до геоінформаційної бази даних

Діаграма дозволяє розробнику робити аналіз кількості запитів, що надходять до бази у вибраний період, на основі цих даних розробник має змогу адаптувати роботи систему та зробити коректний розподіл запитів, що надсилаються до бази даних.

Для перегляду даних, що містить база даних користувачу необхідно зайти у пункт “Data”. У цій вкладці користувач має можливість побачити дані що зберігаються у таблиці у зручному форматі, де відображаються атрибути таблиці та кортежі з даними (рисунок 5.6).

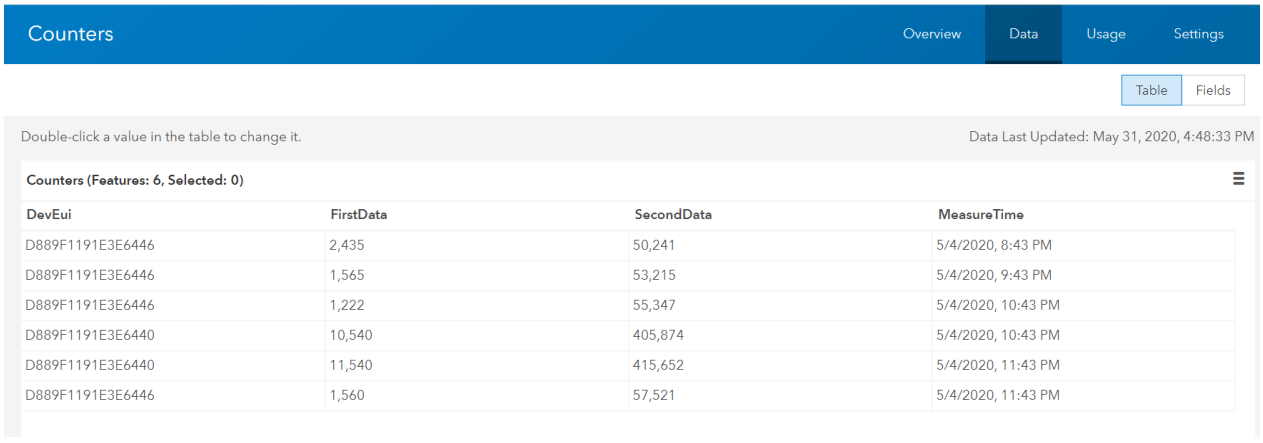

Рисунок 5.6 – Інтерфейс перегляду даних таблиці

Також користувач, що має права вносити зміни до інформаційної бази даних, має можливість редагування, виділення та додавання нових атрибутів до таблиці (рисунок 5.7).

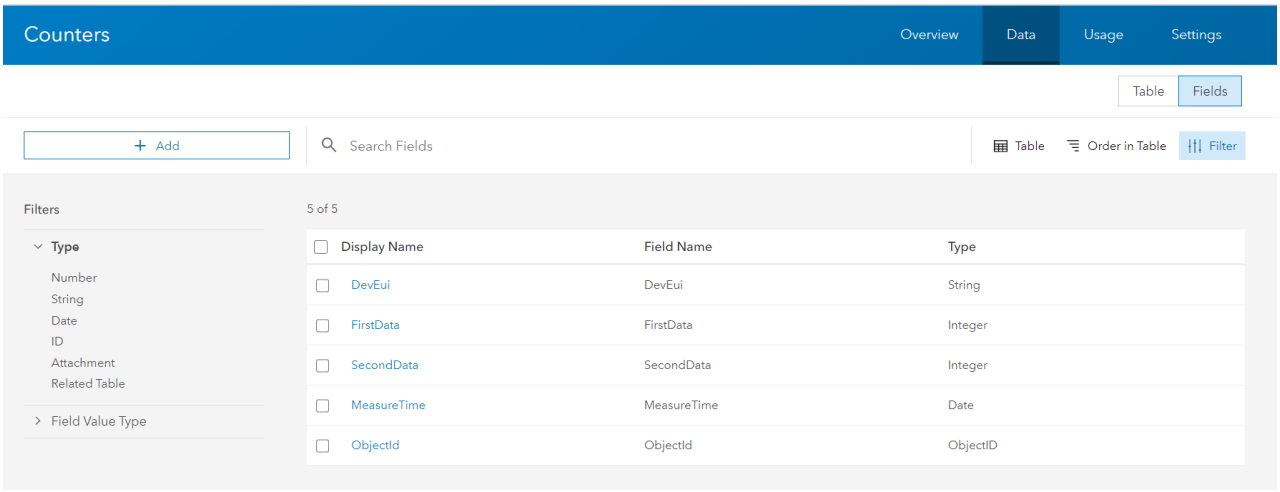

Рисунок 5.7 – Інтерфейс для взаємодії з атрибутами таблиці

Адаптивний інтерфейс також дозволяє переглядати інформацію на мобільних пристроях. Надані інтерфейси для роботи з геоінформаційною базою даних допомагають користувачам та розробнику системи роботи контроль та аналіз даних, що наявні у системі. Процес взаємодії між розробником та веб-застосунком дозволяє аналізувати навантаження на базу даних та переглядати останні записи, які наявні в таблиці. Розробник також має функціонал для створення нових таблиць в базі даних, у випадках розширення або додавання нових об’єктів для обліку інформації.

5.3 Перегляд показників лічильників у Web-інтерфейсі

Для зберігання даних використовуються таблиці геоінформаційної бази даних, що містять інформацію про показники лічильників. Доступ до інформації про показники користувач може отримати із таблиці контенту або через посилання, якщо відкрите діалогове вікно з інформацією про розумний лічильник.

На (рисунку 5.8) зображено вигляд таблиці з даними, що зберігає інформацію про показники, час та одиниці вимірювання лічильника. У деяких полях міститься зв’язана інформація, що позначається символом замку у замкнутому стані.

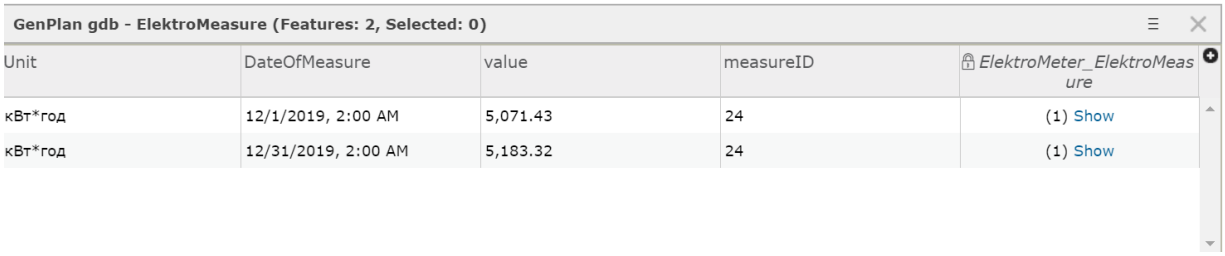

Рисунок 5.8 – Графічне відображення даних з таблиці

У випадку коли користувач натисне на поле “Show”, це дозволяє переглянути інформацію про розумний лічильник та його розміщення на мапі. Можливості програмного забезпечення ArcGIS Online дозволяють створювати власні віджети, тому користувач має змогу розробити власне рішення для аналізу даних з таблиць та їх відображення, що надає перевагу у використанні хмарних технологій для доступу до даних.

ВИСНОВКИ

Під час проходження практики було вирішено задачу з розробки системи для збору, накопиченню та відображення даних вимірювань в інженерних мережах Відповідно до задачі було проаналізовано предметну область з управління розумними датчиками, розглянуто важливі принципи та підходи до проектування та реалізації подібних систем. З метою покращення системи було використано програмні продукти ArcGIS Online та Actility ThingPark Wireless. Було проведено огляд існуючих систем для вирішення задач з якими стикаються системи у сфері Інтернету речей. На основі отриманих знань було спроектовано поточну геоінформаційну систему, що дозволяє користувачам аналізувати дані з розумних лічильників та управляти інженерними мережами за допомогою Web-інтерфейсу.

Для створення ефективної програмної реалізації було досліджено ряд технологій та засобів розробки, обрано типовий стек технологій для розробки геоінформаційної системи. Серед обраних технологій, мов та засобів розробки можна виділити мову програмування JavaScript, фреймворк NestJS, платформу ArcGIS Online, середу розробки JetBrains Webstorm. Під час реалізації програмної системи було отримано практичні навички з використання вказаних технологій, мов програмування та засобів розробки.

Результатом розробки системи є набір інтегрованих між собою платформ, що реалізують функціонал системи для збору, накопиченню та відображення даних вимірювань в інженерних мережах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Internet of Things (IoT): A vision, architectural elements, and security issues. // 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC) / , 2017. – (IEEE Xplore). – С. 492 – 496.
2. Сербенюк С.Н. Картография и Геоинформатика - их взаимодействие / Под ред. В.А. Садовничего. — М.: Изд-во Моск. ун-та, 1990. — 159 с.
3. Геоинформационные системы как эффективный инструмент поддержки экологических исследований. / Солнцев Л.А. // Нижний Новгород: Нижегородский госуниверситет – 2012. – 54с
4. Геоинформационные системы. Учебное пособие для вузов / Бугаевский Л. М., Цветков В. Я. – 2000. – 222 с
5. Moustafa E. Smart Metering Technology and Services - Inspirations for Energy Utilities / Eissa Moustafa., 2016. – 168 с. – (InTechOpen).
6. LoRaWAN™ Specification [Електронний ресурс] / LoRa Alliance. – 2017. – Режим доступу до ресурсу: https://lora-alliance.org/sites/default/files/2018-04/lorawantm_specification_-v1.1.pdf.
7. A Study of LoRa: Long Range & Low Power Networks for the Internet of Things / A. Augustin, J. Yi, T. Clausen, W. Townsley. – 2016.
8. About LoRa Alliance® [Електронний ресурс] – Режим доступу до ресурсу: https://lora-alliance.org/about-lora-alliance.
9. Build powerful apps with the ArcGIS REST API [Електронний ресурс] – Режим доступу до ресурсу: https://developers.arcgis.com/rest/.
10. Инженерные сети, оборудование зданий и сооружений / Е. Н. Бухаркин, В. М. Овсянников, К. С. Орлов и др. / под ред. Ю. П. Соснина. – Москва: Высш. шк., 2001. – 415 с.

11. What is Postman and why use it [Електронний ресурс] – Режим доступу: https://www.digitalcrafts.com/blog/student-blog-what-postman-and-whyuse-it.

12. About Postman [Електронний ресурс] – Режим доступу: https://www.getpostman.com/about-postman.

13. Макконнелл С. Совершенный код. Мастер-класс / Стив Макконнелл. – Русская Редакция: Вильямс, 2013. – 896 с.

Додаток 1

Програмні засоби збору, накопичення та WEB відображення даних вимірювань в інженерних мережах

Специфікація

УКР.НТУУ“КПІ”. ТМ62192_20Б

Аркушів 2

2020

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ«КПІ ім. Ігоря Сікорського». ТМ62192_20Б 81-1	Записка	Пояснювальна записка
Компоненти		
УКР.НТУУ«КПІ». ТМ62192_20Б 12-1	Текст програмного модулю	
УКР.НТУУ«КПІ». ТМ62192_20Б 13-1	Опис програми	

Додаток 2

Програмні засоби збору, накопичення та WEB відображення даних вимірювань в інженерних мережах

Текст програмного модулю

УКР.НТУУ“КПІ”. ТМ62192_20Б 12-1

Аркушів 8

2020

```
import { BadRequestException, Body, Controller, Delete, Post, Query } from
'@nestjs/common';
import { CountersService } from './counters.service';
import { UplinkDto } from './dto/uplink.dto';
import { ArcgisService } from '../arcgis/arcgis.service';
import { MeasureDto } from './dto/measure.dto';

@Controller('counters')
export class CountersController {
  constructor(
    private countersService: CountersService,
    private arcgisService: ArcgisService,
  ) {}

  @Post()
  async createMeasureData(@Body() body: any) {
    if (body.hasOwnProperty('DevEUI_uplink')) {
      const {
        payload_hex: payloadHex,
        DevEUI,
      } = body.DevEUI_uplink as UplinkDto;
      const measurePayload = this.countersService.encodeInformation(payloadHex);
      const measure = new MeasureDto(
        DevEUI,
        measurePayload.periodValue,
        measurePayload.totalValue,
        measurePayload.time,
      );
      return this.arcgisService.sendData(measure.toArcgisMeasure());
    } else {
      throw new BadRequestException('Invalid body without DevEUI_uplink field');
    }
  }

  @Delete()
  async deleteMeasureData(@Query('ids') objectIds: string) {
    const ids = objectIds.split(',');
    return this.arcgisService.removeData(ids);
  }
}

import { Test, TestingModule } from '@nestjs/testing';
import { CountersController } from './counters.controller';
import { CountersService } from './counters.service';
import { ArcgisModule } from '../arcgis/arcgis.module';
import { BadRequestException } from '@nestjs/common';
import { ArcgisService } from '../arcgis/arcgis.service';

describe('CountersController', () => {
  let countersController: CountersController;
  let arcgisService: ArcgisService;

  const bodyMock = {
    'DevEUI_uplink': {
      'Time': '2020-05-04T19:49:52.797+02:00',
      'DevEUI': 'D889F1191E3E6446',
      'payload_hex': '78a7280003d7104326b50a00698207',
      'DevLAT': 50.43969,
      'DevLON': 30.447603,
```

```
      'BatteryLevel': 255,
    },
  };

  beforeEach(async () => {
    const app: TestingModule = await Test.createTestingModule({
      imports: [ ArcgisModule ],
      controllers: [ CountersController ],
      providers: [ CountersService ],
    }).compile();

    countersController = app.get<CountersController>(CountersController);
    arcgisService = app.get<ArcgisService>(ArcgisService);
    arcgisService.sendData = jest.fn();
    arcgisService.removeData = jest.fn();
  });

  afterAll(async () => {
    jest.restoreAllMocks();
  });

  describe('createMeasureData', () => {
    it('should throw error', async () => {
      await expect(countersController.createMeasureData({ val: 'val'
})).rejects.toThrow(BadRequestException);
    });
    it('should call ArcGISService', async () => {
      await countersController.createMeasureData(bodyMock);
      expect(arcgisService.sendData).toHaveBeenCalled();
    });
  });

  describe('deleteMeasureData', () => {
    it('should call ArcGISService', async () => {
      await countersController.deleteMeasureData('1,3,4');
      expect(arcgisService.removeData).toHaveBeenCalled();
    });
  });
});

import { BadRequestException, Injectable } from '@nestjs/common';
import { PayloadDto } from './dto/payload.dto';

@Injectable()
export class CountersService {
  encodeInformation(payloadHex: string): PayloadDto {
    const ID = payloadHex.slice(0, 2);
    if (ID !== '78') {
      throw new BadRequestException('Incorrect payload format');
    }
    const SEQ = Number.parseInt(payloadHex.slice(2, 4), 16);
    const Flag = Number.parseInt(payloadHex.slice(4, 6), 16);
    const payloadHeader = Number.parseInt(payloadHex.slice(6, 8), 16);
    const payloadFlag = Number.parseInt(payloadHex.slice(8, 10), 16);
    const payloadTime = this.formatTime(payloadHex.slice(10, 18));
    const payloadFirstData = this.formatData(payloadHex.slice(18, 24));
    const payloadSecondData = this.formatData(payloadHex.slice(24, 30));

    console.info({
```

```
      SEQ,
      Flag,
      payloadHeader,
      payloadFlag,
      payloadTime,
      payloadFirstData,
      payloadSecondData,
    });

    return {
      sequence: SEQ,
      periodValue: payloadFirstData,
      totalValue: payloadSecondData,
      time: payloadTime,
    };
  }

  formatData(payload: string): number {
    const revertedBytes = this.revertBytes(payload);
    return Number.parseInt(revertedBytes, 16);
  }

  formatTime(payload: string): number {
    const revertedBytes = this.revertBytes(payload);
    const unixTimestamp = Number.parseInt(revertedBytes, 16) + 946684800;
    return unixTimestamp * 1000;
  }

  revertBytes(str: string): string {
    const splitByBytePayload = this.extendedSplit(str, 2);
    return splitByBytePayload.reverse().join('');
  }

  extendedSplit(input: string, length: number): string[] {
    return input.match(
      new RegExp(
        '.{1,' + length + '}(?=(.{' + length + '})+(?!.))|.{1,' + length + '}$',
        'g',
      ),
    );
  }
}

import { Test } from '@nestjs/testing';
import { CountersService } from './counters.service';
import { BadRequestException } from '@nestjs/common';

describe('CountersService', () => {
  let service: CountersService;

  beforeEach(async () => {
    const module = await Test.createTestingModule({
      providers: [ CountersService ],
    }).compile();

    service = module.get<CountersService>(CountersService);
  });

  afterEach(async () => {
```

```
    jest.restoreAllMocks();
  });

  describe('extendedSplit', () => {
    it('should slit string in array with two elements', () => {
      expect(service.extendedSplit('abcd', 2)).toStrictEqual([ 'ab', 'cd' ]);
    });
    it('should slit string in array with three elements', () => {
      expect(service.extendedSplit('abcdefabc', 3)).toStrictEqual([ 'abc', 'def', 'abc'
]);
    });
  });

  describe('revertBytes', () => {
    it('should revert bytes', () => {
      jest.spyOn(service, 'extendedSplit');
      const result = service.revertBytes('abcdef');

      expect(service.extendedSplit).toHaveBeenCalled();
      expect(result).toBe('efcdab');
    });
  });

  describe('formatTime', () => {
    it('should return time in readable format', () => {
      jest.spyOn(service, 'revertBytes');
      const result = service.formatTime('d7104326');

      expect(service.revertBytes).toHaveBeenCalled();
      expect(result).toBe(1588614231000);
    });
  });

  describe('formatData', () => {
    it('should return data in readable format', () => {
      jest.spyOn(service, 'revertBytes');
      const result = service.formatData('b50a00');

      expect(service.revertBytes).toHaveBeenCalled();
      expect(result).toBe(2741);
    });
  });

  describe('encodeInformation', () => {
    it('should throw error if payload have incorrect format', () => {
      const mockPayload = '65a7280003d7104326b50a00698207';
      expect(() =>
service.encodeInformation(mockPayload)).toThrow(BadRequestException);
    });
    it('should return measure', () => {
      const mockPayload = '78a7280003d7104326b50a00698207';
      expect(service.encodeInformation(mockPayload)).toEqual({
        sequence: 167,
        periodValue: 2741,
        totalValue: 492137,
        time: 1588614231000,
      })
    });
  });
```

```
});

import { ArcgisMeasure } from '../../arcgis/interfaces/arcgis-measure';

export class MeasureDto {
  counterID: string;
  periodValue: number;
  totalValue: number;
  time: number;

  constructor(
    counterID: string,
    periodValue: number,
    totalValue: number,
    time: number,
  ) {
    this.counterID = counterID;
    this.periodValue = periodValue;
    this.totalValue = totalValue;
    this.time = time;
  }

  public toArcgisMeasure(): ArcgisMeasure {
    return {
      DevEui: this.counterID,
      FirstData: this.periodValue,
      SecondData: this.totalValue,
      MeasureTime: new Date(this.time),
    };
  }
}

export class PayloadDto {
  sequence: number;
  periodValue: number;
  totalValue: number;
  time: number;
}

export class UplinkDto {
  Time: string;
  DevEUI: string;
  payload_hex: string;
  DevLAT: string;
  DevLON: string;
}

export interface ArcgisMeasure {
  DevEui: string;
  FirstData: number;
  SecondData: number;
  MeasureTime: Date;
}

import { Injectable } from '@nestjs/common';
import { ApplicationSession } from '@esri/arcgis-rest-auth';
import { ConfigService } from '@nestjs/config';
import { applyEdits } from '@esri/arcgis-rest-feature-layer';
import { ArcgisMeasure } from './interfaces/arcgis-measure';
```

```
@Injectable()
export class ArcgisService {
  private readonly serviceUrl: string;
  private readonly authentication: ApplicationSession;

  constructor(private configService: ConfigService) {
    const clientId = this.configService.get<string>('arcgis.clientId');
    const clientSecret = this.configService.get<string>('arcgis.clientSecret');
    this.serviceUrl = this.configService.get<string>('arcgis.serviceUrl');
    this.authentication = new ApplicationSession({
      clientId,
      clientSecret,
    });
  }

  async sendData(measure: ArcgisMeasure) {
    const res = await applyEdits({
      url: this.serviceUrl,
      adds: [
        {
          attributes: {
            ...measure,
          },
        },
      ],
      authentication: this.authentication,
    });
    return res.addResults;
  }

  async removeData(ids: string[]) {
    const res = await applyEdits({
      url: this.serviceUrl,
      deletes: ids,
      authentication: this.authentication,
    });
    return res.deleteResults;
  }
}

import { Test } from '@nestjs/testing';
import { ArcgisService } from './arcgis.service';
import { applyEdits } from '@esri/arcgis-rest-feature-layer';
import { ArcgisMeasure } from './interfaces/arcgis-measure';
import { ConfigService } from '@nestjs/config';

jest.mock('@esri/arcgis-rest-feature-layer', () => ({
  applyEdits: jest.fn().mockReturnValue({
    addResults: {},
    deleteResults: {}
  }),
}));

describe('ArcgisService', () => {
  let service: ArcgisService;
  let mockMeasure: ArcgisMeasure;

  beforeEach(async () => {
```

```
    const module = await Test.createTestingModule({
      providers: [
        ArcgisService,
        {
          provide: ConfigService,
          useValue: {
            get: jest.fn().mockImplementation(str => str),
          },
        },
      ],
    }).compile();

    mockMeasure = {
      DevEui: 'D889F1191E3E6446',
      FirstData: 2435,
      SecondData: 50241,
      MeasureTime: new Date(1588614231000),
    };
    service = module.get<ArcgisService>(ArcgisService);
  });

  afterEach(async () => {
    jest.restoreAllMocks();
  });

  describe('sendData', () => {
    it('should call applyEdits', async () => {
      await service.sendData(mockMeasure);
      expect(applyEdits).toHaveBeenCalled();
    });

    it('should call with Url from config', async () => {
      await service.sendData(mockMeasure);
      expect(applyEdits).toHaveBeenCalledWith(expect.objectContaining({
        url: 'arcgis.serviceUrl'
      }));
    });

    it('should add measure data to request object', async () => {
      await service.sendData(mockMeasure);
      expect(applyEdits).toHaveBeenCalledWith(expect.objectContaining({
        adds: [
          {
            attributes: {
              ...mockMeasure,
            },
          },
        ]
      }));
    });
  });
});
```

Додаток 3

Програмні засоби збору, накопичення та WEB відображення даних вимірювань в інженерних мережах

Опис програмного модулю

УКР.НТУУ“КПІ”. ТМ62192_20Б 13-1

Аркушів 6

2020

АНОТАЦІЯ

Розроблений програмний продукт для збору, накопиченню та WEB відображення являє собою систему, що має серверний застосунок та веб-інтерфейс для відображення та управління даними в інженерних мережах. Система надає доступ користувачам до редагування даних, які надійшли від лічильників, перегляду статистики використання геоінформаційної бази даних та функціонал для передачі даних до веб-інтерфейсу ArcGIS Online.

Користувачами системи можуть бути люди, які мають девайс з доступом до глобальної мережі Інтернет.

ЗМІСТ

1. ВІДОМОСТІ ПРО ПРОГРАМНИЙ МОДУЛЬ

При створенні програмного забезпечення були використані такі засоби реалізації:

— середовище розробки Jetbrains Webstorm, адже має найбільше функціоналу серед конкурентів;

— мову програмування TypeScript для написання серверної частини;

— фреймворк Nest.js, для організації архітектури та взаємодії серверної частини, що забезпечує передачу даних від лічильника до ArcGIS Online;

— платформу Node.js для виконання серверного застосунку;

— програмне забезпечення Postman для симуляції запитів у мережі;

— хмарне середовище ArcGIS Online та сервіс ArcGIS REST API;

— пакетний менеджер NPM для підключення всіх модулів та бібліотек для роботи проекту;

— Git для впровадження системи контролю версій.

1.1. Опис логічної структури

Для реалізації задачі з розробки системи для збору, накопиченню та WEB відображення даних вимірювань в інженерних мережах було розроблено програмний продукт у вигляді серверного застосунку для передачі даних з лічильників, а також створений WEB інтерфейс в системі ArcGIS Online, що надає можливості для зручного перегляду накопиченої інформації та атрибутів даних.

Для взаємодії серверного застосунку, що обробляє дані надіслані датчиками та WEB-інтерфейсу хмарного застосунку на ArcGIS Online, система також використовує платформу ArcGIS REST API, що дозволяє маніпулювати даними та має широкий функціонал на платформі ArcGIS.

1.2. Вхідні та вихідні дані

Вхідними даними для системи є дані, що надсилають лічильники з основними параметрами та показниками. Формат даних та кількість інформації, що надходить до системи регулюється виробником лічильників та модулів, що можуть бути встановлені на них.

Вихідними даними є геоінформаційна база даних та мапа на якій зображено інженерні мережі та розташування лічильників. Геоінформаційна база даних містить інформацію про показники, що лічильник надіслав до системи. Можливість перегляду карти дозволяє користувачам візуально побачити місцеположення датчиків та редагувати основні атрибути таблиць.

2. ВИКОРИСТАНІ ТЕХНІЧНІ ЗАСОБИ

Система була протестована в браузері Google Chrome версії 83.0.4103.97 на персональному комп’ютері, який працює на операційній системі Windows 10 на базі процесора x64 Intel Core i5 (8th Gen) та має 8 Гб оперативної пам’яті. Для запуску серверного застосунку було встановлено платформу Node.js версії 12.16.1 та менеджер пакетів версії 6.14.4. Для відображення інтерфейсу системи необхідні права доступу до аплікації у хмарному середовищі ArcGIS Online.